MASTER EN SCIENCES INFORMATIQUES

Étude des techniques de recommandation et de contextualisation pour la conception d'une application touristique mobile

Lejeune, M; Chevalier, S

Award date:
2012

Awarding institution:
Universite de Namur

Facultés Universitaires Notre-Dame de la Paix, Namur
Faculté d'Informatique
Année académique 2011–2012

Étude des techniques de recommandation et de contextualisation pour la conception d'une application touristique mobile

Stéphane Chevalier Martin Lejeune

Mémoire présenté en vue de l'obtention du grade de
Master en Sciences Informatiques

Résumé

L'expérience du touriste pourrait être améliorée par les nouvelles technologies. Son smartphone a maintenant la capacité technique de le guider et de le conseiller dans ses activités. Dans ce cadre, les différentes techniques de recommandation traditionnelles *content-based* et collaboratives ainsi que les techniques de recommandations contextualisées ont été analysées et comparées. Ce travail propose également l'architecture d'un système mobile de recommandation contextualisé à destination des touristes en fonction de leurs comportements et des problèmes qu'ils rencontrent. Cette architecture présente des solutions afin de représenter le contexte dans lequel les visites touristiques sont réalisées. Parmi différentes techniques de modélisation de contexte, l'utilisation d'ontologies apparait comme la plus appropriée. Dès lors, une ontologie est créée et des règles d'inférence sont définies. Enfin, un prototype est développé et présenté.

Mots-clés : systèmes de recommandation, systèmes contextualisés, ontologies, tourisme, mobile, prototype

Abstract

New technologies could enhance the tourist's experience. His smartphone is now able to advise and guide him. Consequently, this work studies the use of recommender and context-aware systems in the field of tourism. The different traditional content-based and collaborative recommender systems as well as context-aware recommendation techniques are analysed and compared. An architecture of a mobile touristic context-aware recommender system is presented, taking into account the tourists' behaviours and the problems encountered before or during their trips. This architecture shows some solutions for representing the context in which a visit takes place. Among different context modeling techniques, the use of ontologies appears as the most appropriate one. Therefore, an ontology is created and inference rules are defined. Finally, a prototype is developed and presented.

Keywords : recommender systems, context-aware systems, ontologies, tourism, mobile, prototype

Avant-propos

Ce mémoire constitue le couronnement de notre cursus au sein des Facultés Universitaires Notre-Dame de la Paix à Namur. C'est pourquoi il aura nécessité une grande partie des compétences développées grâce à notre parcours universitaire. Nous tenons donc premièrement à remercier l'ensemble du corps professoral et administratif de cette institution, ainsi que tous nos collègues étudiants pour l'apprentissage mutuel tiré de nos collaborations.

En deuxième lieu, nous remercions notre promoteur, le Pr Philippe Thiran, d'avoir supervisé notre mémoire et de nous avoir donné la possibilité unique de nous rendre au Brésil pour l'accomplissement de notre stage.

Troisièmement, nous remercions chaleureusement le Pr Leandro Krug Wives de l'Universidade Federal do Rio Grande do Sul, à Porto Alegre, Brésil, pour son franc soutien durant le déroulement de notre stage. Sa disponibilité, ses connaissances et son enthousiasme ont transformé notre séjour en une expérience hautement enrichissante, aussi bien du point de vue académique qu'humain.

Nous remercions aussi toutes les autres personnes rencontrées au Brésil de nous avoir accueillis, guidés, accompagnés durant ces trois mois. Il ne fait aucun doute que leur rencontre fut une chance et qu'elle aura grandement contribué à l'excellent souvenir que nous laisse cette période.

Nous remercions également nos familles, pour les nombreuses heures passées à nous conseiller, ou à relire et corriger ce document.

Enfin, nous remercions l'ensemble du jury de porter attention à notre travail, et nous leur souhaitons une bonne lecture.

Table des matières

Chapitre 1

Introduction

1.1 Contexte

1.1.1 Le secteur touristique

C'est un fait, le tourisme représente aujourd'hui l'une des industries les plus prospères dans le monde. Selon l'Organisation Mondiale du Tourisme, en 2010 les recettes du tourisme ont atteint les 693 milliards d'euros. Ce chiffre impressionnant représente une augmentation de 4,7% par rapport à l'année précédente, continuant ainsi une hausse des recettes significative, année après année.

Le secteur touristique est l'un des rares à ne pas souffrir de la crise économique, débutée en 2008. Mieux encore, il permet à certains pays de résister à une économie vacillante. Ainsi, le tourisme en Espagne offre actuellement 2.5 millions d'emplois et constitue un véritable pilier économique de ce pays, dont le taux de chômage dépasse aujourd'hui les 24% [32].

Le tourisme n'a donc pratiquement pas connu la crise économique, sa reprise ayant débuté dès 2010 avec une augmentation de 6.7% d'arrivées des touristes internationaux, alors que les prévisionnistes les plus optimistes estimaient une amélioration ne dépassant pas les 5%. Dans le monde, le Bureau International du Travail estime que pas moins de 61 millions d'emplois seront créés dans ce secteur d'ici 2019 [5]. L'explication est simple : les vacances sont perçues comme

FIGURE *1.1 – Évolution du tourisme international. Source : World Tourism Organization (UNWTO)*

indispensables par beaucoup, et occupent désormais une place importante dans le budget annuel de nombreux foyers [25].

Néanmoins, mis à part pour le choix et l'achat de services touristiques, les technologies actuelles n'offrent au touriste qu'une aide élémentaire. En effet, le touriste à la recherche de support à son expérience touristique se heurte vite au manque criant de systèmes modernes adaptés. Preuve en est que le touriste du 21e siècle est souvent croisé avec un épais guide à la main. Pourtant, les évolutions techniques de ces dernières décennies pourraient offrir au voyageur de loisir une expérience d'un tout autre niveau. Premièrement, on constate une démocratisation rapide des appareils mobiles. Ceux-ci, continuellement plus rapides et polyvalents, ont déjà prouvé leur efficacité dans de nombreux domaines, tels que celui de la navigation GPS. Deuxièmement, la connectivité mobile ne cesse de s'améliorer, au travers des réseaux EDGE, 3G et LTE. Si l'accès à ceux-ci reste encore relativement coûteux, des offres plus accessibles sont développées par les opérateurs, tendant vers « l'internet mobile pour tous ».

Dans le domaine scientifique, plusieurs recherches récentes ont abordé ce sujet. Parmi elles, certaines ont donné lieu à des prototypes. Néanmoins, aucune solution ne semble avoir fait

Smartphone Installed Audience (000)

Source: comScore MobiLens, 3 mon. avg. ending Dec-2011

UK	25,386
Spain	17,855
Canada	9,103
Italy	21,067
US	97,865
France	18,788
Germany	21,300
Japan	16,902

FIGURE *1.2 – Nombre de smartphones en circulation, en millions. Source : comScore MobiLens*

l'unanimité, et la communauté peine encore à fixer des standards dans ce domaine. En cause, le manque d'universalité de la plupart des solutions, souvent limitées à un fonctionnement dans une zone géographique restreinte. De plus, de nombreux prototypes embarquent des systèmes de recommandations posant de gros problèmes de performances lors de tests à grande échelle [13].

En conclusion, le domaine possède donc un important potentiel du point de vue de la recherche, nous poussons à nous y intéresser plus en avant.

1.1.2 La démocratisation des smartphones

Depuis le lancement du célèbre iPhone d'Apple en 2007, les lancements et les ventes de smartphones ne cessent d'augmenter. En décembre 2011, on dénombrait plus de 98 millions de téléphones intelligents en circulation aux États-Unis. La figure 1.2 (p.3) illustre le nombre de smartphones en circulation dans les différents pays européens. L'apparition de ces nouveaux appareils a été accompagnée de la naissance de plusieurs systèmes d'exploitation mobiles. Les plus répandus : iOS (Apple) et Android (Google). Ainsi, la figure 1.3 (p.4) nous renseigne sur les différentes parts de marché. À eux seuls, iOS et Android équipent plus de 80% des smartphones, au détriment d'autres géants tels que Microsoft, Nokia et RIM (père des téléphones Blackberry)[4].

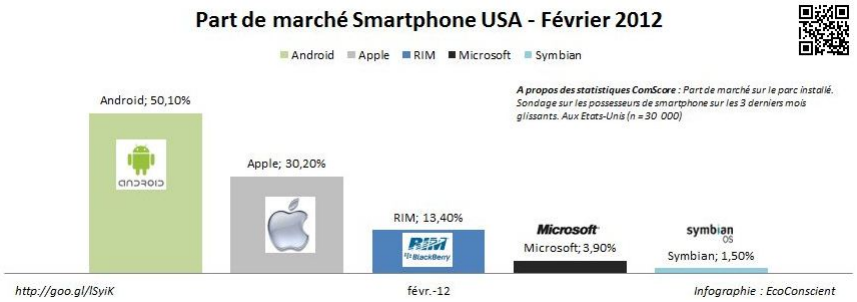

FIGURE *1.3 – Part de marché des systèmes d'exploitation. Source : EcoConscient*

Cet oligopole offre cependant un précieux avantage pour les développeurs d'applications mobiles. En effet, chaque plate-forme ne demande la maîtrise que d'un seul langage de programmation (Objective-C pour iOS, et un dérivé de Java pour Android). De plus, des outils commencent à apparaitre afin d'automatiser au maximum le portage vers le système concurrent. Il est donc aujourd'hui aisé de toucher à la fois les clients d'Apple et ceux de Google.

Justement, les applications mobiles sont à la mode et le nombre de développeurs dans ce domaine augmente continuellement. Ainsi, le développement d'une application mobile fait partie de la stratégie d'une part croissante de sociétés dont l'image publique revêt une importance cruciale.

L'évolution des marchés d'applications durant l'année 2011 est impressionnante. L'Android Market (aujourd'hui rebaptisé Google Play) fait preuve d'une croissance record en doublant de taille et possède désormais plus de 400 000 applications. Aujourd'hui, on estime qu'environ 2000 nouvelles applications sont créées quotidiennement.

Pendant longtemps, Apple a dominé le marché des systèmes d'exploitation mobiles. Néanmoins, si le gouffre entre les deux leaders s'est résorbé, Apple possède encore un léger avantage. En effet, 59% des applications développées sont à destination de l'AppStore. Ce choix s'explique essentiellement par le profil des utilisateurs d'iPhone, qui sont généralement plus enclins à débourser régulièrement quelques euros pour de nouvelles applications.

À ce jour, l'AppStore a rapporté près de 5 milliards de dollars alors que les recettes engen-

drées par l'Android Market se situent autour des 350 millions de dollars. Cette grande différence qui s'explique aussi par une plus grande présence d'applications gratuites sur ce dernier.

Nous pouvons conclure en déduisant que le choix de la première plate-forme s'effectue souvent selon le type d'application. Des développeurs inconnus souhaitant mettre à disposition une application sans prétention auront tendance à s'orienter vers le système Android. Inversement, une personne ou une société à la recherche de rentabilité financière aura une préférence pour être présente sur l'iPhone d'Apple.

1.2 Motivations

Malgré les recherches scientifiques dans le domaine des applications touristiques mobiles, peu sont exploitées et utilisées par le grand public. Pourtant, une application mobile efficace pourrait apporter une aide précieuse à une personne se situant dans un environnement qui lui est étranger.

En effet, il est actuellement difficile de se procurer une application équipée d'un système de recommandation satisfaisant à ses besoins. Par exemple, celles-ci possèdent généralement un caractère trop dirigiste qui ne plait pas à l'utilisateur, souhaitant rester maitre de la planification de son séjour. D'autres problèmes récurrents rencontrés dans les applications actuelles seront abordés dans la suite de ce document.

Les comportements touristiques sont bien ancrés dans les habitudes des visiteurs et sont difficiles à modifier. Lire le guide d'une ville, chercher son chemin, demander des informations et se renseigner dans les offices du tourisme sont autant d'activités qui pour beaucoup, font partie intégrante de leurs séjours à l'étranger. Les applications visant à supprimer ces aspects sont donc actuellement mal perçues par les utilisateurs, et restent d'utilisation confidentielle [16][3].

Pourtant, les touristes sont confrontés à un problème majeur de choix. En effet, le touriste a rarement le temps de pouvoir visiter l'ensemble des points d'intérêts présents dans la région et

se retrouve souvent avec une quantité d'informations considérable à gérer, que ce soit à travers les guides, la publicité régionale, internet... Choisir ses activités, sachant qu'il ne retournera probablement pas dans la région, est donc rarement une chose aisée.

1.3 Objectifs

Le but de ce travail est de fournir un système fonctionnel rassemblant différents paradigmes susceptibles d'améliorer l'expérience touristique. Premièrement, l'utilisation d'un système de recommandation permet de fournir au touriste des propositions de points d'intérêts personnalisés sur base de son profil et de son comportement. De cette façon, l'utilisateur n'est pas noyé sous une masse d'informations d'intérêt variable. Ensuite, la contextualisation du système. Au travers de la prise en compte de la météo, de l'heure du jour, de l'appareil de l'utilisateur... Elle offre une expérience d'utilisation plus agréable tout en améliorant la précision des informations fournies.

Contrairement à d'autres solutions, dont certaines, comme COMPASS, seront présentées dans le chapitre 3 (p.33) , l'objectif est de proposer au touriste un support peu invasif lors de ses visites. En effet, des tests en conditions réelles ont mis en exergue des retours négatifs des utilisateurs face à des systèmes « trop intelligents » [17]. Plus précisément, le touriste ne souhaite pas qu'une machine supplée à son jugement concernant la pertinence d'une information [43].

Pour ce faire, notre projet s'abstient d'occulter les points d'intérêts potentiellement moins adéquats et préfère proposer une classification des différentes possibilités. Dans le même esprit, nous désirons laisser à l'utilisateur une liberté totale dans la planification de son itinéraire. Notre proposition vise donc un compromis entre une assistance personnalisée et l'autonomie du touriste.

Finalement, l'enjeu ici est de proposer une application peu gourmande en ressources, évitant ainsi les écueils d'autres prototypes. Nous tenterons de mettre au point un système qui, selon nous, n'enlève en rien le sentiment d'aventure et de liberté de son utilisateur, tout en fournissant une quantité précieuse d'informations pour son parcours touristique, où qu'il soit.

1.4 Méthodologie scientifique et plan

Ce travail est composé en plusieurs parties, dont l'ordre correspond à notre processus de recherche et développement d'un système de recommandation contextualisé touristique.

Le chapitre suivant détaille les différentes technologies de recommandation et de contextualisation. Nous les définirons et verrons également leurs avantages et inconvénients. Dans la suite de ce travail, certaines des techniques présentées seront employées dans l'architecture du système présenté.

Le chapitre 3 (p.33) concerne l'état de l'art. Nous parlerons des recherches scientifiques et des projets existants portant sur les systèmes de recommandation et de contextualisation dans le milieu touristique. Par ce tour d'horizon, nous constaterons les idées ayant fonctionné, mais aussi celles boudées par les utilisateurs. Les problèmes récurrents dans le développement de ce genre d'applications pourront également être relevés.

Par la suite, il sera question dans le chapitre 4 (p.59) de l'identification des utilisateurs potentiels. Nous y aborderons les comportements touristiques et les problèmes pouvant être rencontrés avant et pendant un séjour à l'étranger. Nous verrons aussi quels sont les outils traditionnels couramment utilisés par les touristes, et comment une application peut se faire une place parmi eux.

Le chapitre 5 (p.67) a pour objectif de présenter l'architecture d'un système de recommandation contextualisé répondant aux problèmes rencontrés par les touristes. Pour cela, nous expliquerons nos choix de méthodes de représentation du contexte et de recommandation.

Nous terminerons ce chapitre par l'architecture du système. Nous en décrirons les différentes parties et commenterons la manière dont elle permet de répondre aux problèmes récurrents des systèmes d'assistance touristique, présentés au chapitre 3 (p.33) .

Afin de valider l'architecture proposée, un prototype, nommé SmartTrip, a été développé et

est présenté dans le chapitre 6 (p.81) . En effet, la construction de ce prototype permet de démontrer la faisabilité du système imaginé dans le chapitre précédent. Nous verrons comment celui-ci a été construit, et quels sont les choix technologiques qui ont pu être effectués.

Nous terminerons ce travail par les conclusions pouvant en être tirées et les perspectives pouvant s'inscrire dans la continuité de cette recherche.

Chapitre 2

Paradigmes

2.1 Introduction

L'objectif de ce chapitre est de définir et détailler les différentes technologies utilisées ou ayant été envisagées durant le développement de notre projet. Ces techniques sont d'utilisation courante dans le monde des applications mobiles.

Nous commencerons en définissant les systèmes contextualisés, et les difficultés qu'ils peuvent rencontrer.

Nous parlerons ensuite des systèmes de recommandation et des différents algorithmes existants. Les problèmes qu'ils peuvent rencontrer, telles les attaques dont ils peuvent être victimes, seront également abordés.

Nous continuerons en étudiant les différentes façons de combiner l'utilisation d'un système de recommandation et d'informations contextuelles dans le but d'améliorer son efficacité.

Enfin, nous aborderons le thème des ontologies. En effet, celles-ci sont couramment utilisées lors de développements de systèmes informatiques portant sur le tourisme. COMPASS, CAMTON et les recherches de Chien-Chih Yu et Hsiao-ping Chang en sont quelques exemples. Nous

verrons notamment leur composition, les langages existants et l'utilité des moteurs d'inférences. Pour finir, nous verrons pourquoi les ontologies sont considérées comme les structures les plus à même de supporter la gestion des informations de contexte.

2.2 Systèmes contextualisés

2.2.1 Introduction

Avec l'apparition des appareils mobiles, les applications sont amenées à fonctionner dans des environnements changeants. Pourtant, rares sont celles qui parviennent à s'adapter efficacement à leurs conditions d'utilisation. Ces dernières années, de nombreux articles ont été publiés dans le but d'améliorer la capture et le traitement automatique des informations de contexte.

2.2.2 Définition

Afin de mieux comprendre les systèmes contextualisés, il est nécessaire de disposer d'une définition correcte du contexte. Dans la littérature, les premiers auteurs utilisant ce terme sont Schilit et Theimer. Ils tentent de le définir par le biais d'exemples [41]. Pour eux, le contexte se réfère à la localisation, l'identification des personnes et intérêts situés à proximité et à leurs changements apportés.

D'après Dey, cette définition est trop spécifique. En effet, le contexte est l'ensemble des aspects d'une situation pertinents pour une application. Il est impossible d'énumérer les aspects considérés comme importants dans toutes les situations, le degré d'importance d'un aspect changeant d'une situation à l'autre.

Pour cette raison, Dey ne se satisfait pas de la définition de Schilit et Theimer [23], et propose :

> *« Context is any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications*

themselves. »

À cela, Dey ajoute quatre catégories d'information de contexte :

- *Identité* : représente le « qui ». Il peut s'agir aussi bien de personnes individuelles que de groupes.
- *Temps*
- *Localisation* : Ne se limite pas à une coordonnée dans un plan en 2D, mais peut aussi intégrer l'orientation, l'altitude...
- *Activité (statut)* : Identifie les caractéristiques intrinsèques de l'entité pouvant être détectées (température ambiante d'un endroit, état de fatigue d'une personne...)

Cette définition est depuis considérée comme une référence en la matière, citée dans la plupart des publications ultérieures.

Schilit et Theimer parlent pour la première fois de logiciels *context-aware* en 1994 [41]. Pour eux, de tels logiciels sont ceux capables de s'adapter en fonction de la localisation de l'utilisateur et de l'ensemble des gens et des objets situés à proximité ainsi que l'évolution de ces entités à travers le temps. Par la suite, de nombreuses tentatives de définition ont été réalisées, sans jamais obtenir le consensus de la communauté scientifique. Selon Dey, celles-ci étaient trop spécifiques, ce qui le poussa à proposer la définition suivante, qui sera elle aussi reconnue comme standard par la suite :

> *« A system is context-aware if it uses context to provide relevant information and/or services to the user, where relevancy depends on the user's task. »*

Dey identifie aussi trois catégories de fonctionnalités que les applications *context-aware* peuvent fournir :

- *Présenter* de l'information et des services à l'utilisateur
- *Exécution automatique* d'un service à un utilisateur
- *Associer le contexte* à l'information pour une utilisation ultérieure

Par la suite, Dey *et coll.* ont conçu Context Toolkit, une architecture visant à faciliter le développement d'applications contextualisées [23].

2.2.3 Les difficultés

L'une des difficultés principales liées aux systèmes contextualisés est la représentation des données de contexte. Néanmoins, l'utilisation d'ontologies s'est progressivement révélée être une technique adaptée à la résolution de ce problème. Ce sujet est traité dans une section ultérieure 2.5 (p.25) .

Ensuite, il est parfois peu aisé de définir une fréquence optimale de mise à jour des données de contexte. En effet, une fréquence trop élevée peut compromettre l'autonomie de l'appareil ou utiliser une part trop importante des ressources. Au contraire, une fréquence trop faible ouvrirait la porte à une représentation biaisée de l'environnement.

De même, l'utilisation de capteurs dans l'environnement a longtemps été un frein pour le développement d'applications à vocation universelle [7]. Toutefois, l'évolution des technologies mobiles a permis de pallier en partie ce problème (par la démocratisation des puces GPS intégrées par exemple).

Enfin, détecter et enregistrer en permanence le contexte d'utilisation d'une application soulève des craintes quant aux risques de violation de la vie privée [24].

2.3 Systèmes de recommandation

2.3.1 Définition

Les systèmes de recommandation sont une technique de filtrage particulière ayant pour objectif de renseigner à son utilisateur des objets (point d'intérêts, livres, films...) potentiellement en adéquation avec ses goûts. Nous pouvons dégager deux types d'utilisation :

- Utilisation de prédiction : l'objectif est de prédire la note qu'un utilisateur donnerait à un certain objet.
- Utilisation de sélection : parmi une liste contenant N objets, le but du système est de pouvoir sélectionner de manière automatique les K objets susceptibles de présenter le plus d'intérêt pour un utilisateur.

De telles pratiques permettent de positionner l'utilisateur au centre du système d'information et la percée des réseaux sociaux a rendu leur utilisation de plus en plus populaire dans le monde du web. Aujourd'hui, les utilisateurs, parfois sans en être conscients, utilisent ces systèmes. En effet, les grands sites web tels que Facebook, Amazon, eBay possèdent aujourd'hui leurs propres systèmes de recommandation.

De plus, ces techniques sont aussi utilisées dans le domaine de la publicité sur internet. La publicité personnalisée est désormais utilisée par la plupart des sites internet de façon à rendre celle-ci plus rentable. Leur contenu est donc rarement anodin et est personnalisé selon les habitudes de navigation du visiteur. Le profil de l'internaute est construit en récoltant ses historiques, souvent à son insu.

Si l'éthique de certains usages peut laisser à désirer, le potentiel de ces systèmes est énorme. S'ils sont bien configurés, la pertinence des résultats peut être excellente. Il est dès lors possible de construire des listes personnalisées pour l'utilisateur, selon ses préférences, quelle que soit la nature des objets.

2.3.2 Techniques

Plusieurs techniques existent, et leurs utilisations varient selon le type de résultat que l'on souhaite obtenir [11].

La première étape importante consiste en la récolte des données sur lesquelles l'algorithme va reposer. Selon le type de données recueillies, nous pouvons séparer les systèmes de personnalisation en deux grandes familles : les systèmes *content-based* et ceux dits collaboratifs.

Systèmes *content-based*

Tout d'abord, il y a ceux qui ne considèrent que les informations appartenant à l'utilisateur en question. Les résultats sont donc produits uniquement à partir de son historique, de son profil et de celui des objets. Typiquement, les profils peuvent consister en un ensemble de mots clés, ou être représentés à l'aide des ontologies. Les profils des utilisateurs sont construits sur base des objets visités, achetés... Et l'objectif ici est clairement la recommandation d'objets similaires. On dira que ces systèmes sont orientés contenus ou *content-based*.

Prenons l'exemple d'un tel système associé à une base de données contenant les descriptions de produits alimentaires vendus par une société. Un client ayant précédemment acheté des biscuits au chocolat et de la mousse au chocolat pourrait se voir recommander d'autres produits contenant du chocolat. En effet, en se basant sur l'historique des achats, le système a pu détecter une corrélation entre les descriptions des différents aliments et a ainsi pu conseiller de nouveaux produits possédant les mêmes caractéristiques.

Par cet exemple, nous pouvons constater le désavantage principal de ces types de systèmes. Si ceux-ci peuvent se révéler efficaces pour proposer des objets similaires, ils éprouvent plus de difficulté pour inclure de la nouveauté dans leurs recommandations et ainsi permettre à l'utilisateur de faire de nouvelles découvertes.

En revanche, en procédant de la sorte, ces systèmes possèdent un avantage considérable : ils peuvent être développés du côté client et ainsi soulager le serveur principal.

Systèmes collaboratifs

De l'autre côté, nous trouvons les systèmes collaboratifs dits *social-based*. La caractéristique de ces systèmes est d'utiliser l'ensemble des données de tous les utilisateurs. On ne se limite plus au simple historique personnel de l'utilisateur. Contrairement aux approches orientées contenu, ces systèmes collaboratifs possèdent donc l'avantage de pouvoir surprendre l'utilisateur en proposant des objets auxquels celui-ci ne s'attendrait pas. Il est dès lors possible de faire de belles découvertes.

Les algorithmes des systèmes collaboratifs varient également selon le type d'informations utilisées. Nous distinguons deux catégories : ceux possédant un algorithme *item-based* (aussi appelés *memory-based*) et ceux possédant un algorithme *user-based* (également connus sous le nom *model-based*).

Algorithmes item-based Les algorithmes commencent par calculer une matrice de similarité des objets. Dans le cas où deux objets sont fréquemment notés de manière similaire par l'ensemble de la communauté, leur degré de similarité augmente.

Par la suite, le système peut recommander à l'utilisateur des objets similaires à son historique. Pour cela, il procède à une technique de moyenne pondérée :

$$P_{u,i} = \frac{\sum_{objetsSimilaires,N}(S_{i,N} * R_{u,N})}{\sum_{objetsSimilaires,N}(|S_{i,N}|)}$$

Dans un premier temps, seuls les objets similaires ayant été précédemment notés par l'utilisateur sont pris en compte. La note de chaque objet est pondérée par le degré de similarité avec l'objet visé. La prédiction est ensuite obtenue en divisant le résultat précédemment obtenu par la somme de toutes les similarités.

Algorithmes *user-based* Ces algorithmes fonctionnent en générant une liste des utilisateurs similaires à l'utilisateur actif, possédant un comportement semblable. Leurs votes permettent de détecter et de représenter le comportement de l'utilisateur. Dans le cas où des utilisateurs similaires ont noté des objets inconnus à l'utilisateur en question, il est désormais possible de le recommander ou d'estimer sa note personnelle.

Chaque famille possède donc ses avantages et ses inconvénients, et leur utilisation dépend essentiellement du type d'application. Il est en effet parfois agréable de pouvoir soulager le serveur, de limiter les communications en développant ces algorithmes sur la plate-forme du client. Un système entièrement développé sur la machine du client peut aussi augmenter sa confiance. Dans ce cas de figure, le respect de la vie privée s'en trouve amélioré. Ajoutons aussi

que cela peut avoir un impact sur les performances de l'application, et que l'objectif n'est pas toujours de surprendre l'utilisateur [11].

Représentation des données

Comme il a été dit précédemment, la qualité des données utilisées est un aspect critique dans ce domaine qui détermine la précision dans les résultats obtenus. Généralement, deux types de données sont exploitées : les données explicites que les utilisateurs doivent renseigner explicitement au système (lorsqu'ils notent des objets, par exemple) et les données implicites. Dans ce cas, il s'agit des informations supplémentaires inférées de manière automatique à partir des données existantes portant sur l'utilisateur.

Les données explicites regroupent les notes données par l'utilisateur, les données d'utilité et de préférence. Dans ce dernier cas, il est sujet des données entrées par l'utilisateur devant renseigner le système sur ses préférences devant certaines situations. Il existe donc deux grandes possibilités de représentation : l'utilisation de nombres ou l'utilisation de mots-clés. Prenons par exemple le cas d'une application chargée de recommander des films à ses utilisateurs. Par la méthode mathématique, ceux-ci pourraient indiquer leurs genres préférés en leur assignant une note. Il est dès lors aisé de déterminer les préférences des utilisateurs.

D'un autre côté, le système pourrait se contenter de demander des mots-clés à son utilisateur. Celui-ci pourrait entrer ses genres préférés, les films récemment appréciés ainsi que ceux détestés...

Les avantages et désavantages de chacune de ces techniques peuvent paraître évidents. Alors que l'utilisation de nombres facilite grandement la tâche du système de recommandation, nous nous heurtons à la subjectivité des utilisateurs. En effet, un 3/5 d'un certain utilisateur peut être équivalent à un 4/5 d'un autre. Tout le monde ne cote pas de la même manière, ce qui peut nuire à la précision des résultats finaux.

Si l'utilisation de mots-clés est plus aisée pour les clients et assure une meilleure objectivité, la tâche du système devant déduire les préférences s'en trouve malheureusement souvent plus ardue [10].

Les données implicites sont générées automatiquement, sans que l'utilisateur en ait conscience. Par exemple, une vidéo jouée en boucle peut impliquer que l'utilisateur l'apprécie. À l'opposé, si celui-ci quitte rapidement une page proposant une autre vidéo, il est possible d'en déduire que celle-ci ne lui semble pas pertinente.

Analyser les interactions et déduire de l'information de manière automatique permet d'obtenir de la consistance parmi l'ensemble des utilisateurs. Le processus fonctionne de la même manière pour tout le monde, et n'est donc soumis à aucune subjectivité. De plus, comme les exemples ci-dessus le montrent, il est possible de catégoriser de manière optimale les interactions ayant un aspect positif et renforçant la préférence pour certains types d'intérêts de celles ayant un aspect négatif.

Enfin, de nombreuses études ont démontré le fait que la plupart des personnes ne sont pas prêtes à dépenser du temps dans la configuration d'un système. Dans la plupart des cas, une application non configurée conduit à des résultats imprécis, et fait perdre du temps à l'utilisateur [19][11].

Il est néanmoins compliqué, voire impossible, de modifier le comportement humain. Au programme de s'adapter, démontrant une fois de plus l'importance cruciale des données implicites et des techniques associées.

2.3.3 Les problèmes majeurs

Problèmes d'insertion

Lorsqu'une personne utilise pour la première fois un système de recommandation, celui-ci ne possède généralement aucun historique. De plus, il a été mentionné précédemment que les utilisateurs n'apprécient guère de paramétrer une application. Rares sont ceux qui le font. Il est donc difficile, voire impossible pour le moteur de recommandation de fonctionner de manière optimale pour cette nouvelle personne. On entre alors dans un cercle vicieux : le système attend

de l'utilisateur qu'il se crée un historique ou qu'il fournisse explicitement de l'information le concernant, mais celui-ci est de moins en moins enclin à le faire puisque les résultats obtenus jusqu'à présent sont généralement trop imprécis pour l'inciter à réutiliser l'application dans le futur. Ce problème est couramment appelé *the new user problem*, soit le problème du nouvel utilisateur [33].

Le problème similaire existe pour l'ajout de nouveaux objets. En effet, un objet récemment inséré n'est apprécié par aucun utilisateur et n'est lié avec aucun autre objet. Il est donc très difficile pour le système de le recommander à ses utilisateurs et on retrouve un cercle vicieux identique à celui lié au problème du nouvel utilisateur. L'objet n'est jamais recommandé parce qu'aucun utilisateur ne lui porte d'intérêt puisqu'il n'est jamais recommandé...

Par ces deux difficultés, on peut aisément comprendre que le lancement de tels systèmes de recommandation peut s'avérer très compliquée. Un système jeune possède peu de données, peu d'utilisateurs et peu de liens entre ceux-ci et les objets à recommander. Il est dès lors difficile de fournir des résultats précis et pertinents. Ce problème récurrent dans le domaine est connu sous le nom de *cold start problem*[40].

Plusieurs techniques ont vu le jour pour tenter de contourner, du moins partiellement, ces difficultés. C'est ainsi que des approches dites hybrides ont vu le jour [14] [34]. De manière générale, ces systèmes hybrides tentent d'utiliser à la fois des algorithmes orientés contenus et des algorithmes utilisant une approche collaborative afin de bénéficier des avantages de chacun selon le contexte de la situation [11].

D'autres possibilités consistent en l'utilisation d'ontologies, ou de bases de données externes afin de forcer la génération de profils et de combler au mieux le manque d'information dans le système.

Difficultés liées aux données

Faible densité des données (problème de rareté) Dans beaucoup de systèmes, le nombre d'objets présents est considérable. Dès lors, même le plus prolifique des utilisateurs ne notera explicitement ou implicitement qu'un très petit pourcentage de tous les objets. Il en résulte que les paires d'utilisateurs n'auront pas ou peu d'objets notés en commun, ce qui rend le calcul des plus proches voisins compliqué et son résultat imprécis.

Le problème du *cold start* détaillé précédemment peut être vu comme un cas spécial du problème de rareté.

Évolution L'évolution des bases de données associées aux systèmes de recommandation peut souvent poser problème concernant les performances. En effet, le système doit être capable de gérer une quantité de données croissant généralement d'une manière très rapide. Il peut par exemple devenir compliqué de trouver les utilisateurs les plus similaires dans une base de données en comprenant des milliers, chacun ayant noté un nombre considérable d'objets différents.

Il n'existe malheureusement pas de solution miracle. La méthode la plus efficace consiste à réduire la taille du problème. Concrètement, cela se traduit en limitant le nombre d'utilisateurs devant être comparés ou la dimension de la matrice de similarité des objets.

Les attaques

Les systèmes de recommandation peuvent subir des attaques d'utilisateurs malicieux. L'objectif est généralement de forcer la recommandation d'un objet particulier auprès d'un groupe ciblé ou de l'ensemble de la communauté. O'Mahony identifie deux catégories d'attaques [36] :

- Attaques de type *push* : le but est d'augmenter la présence de l'objet ciblé en améliorant sa note générale. Pour ce faire, la personne crée un grand nombre de profils, et note l'objet de manière très positive. Celui-ci possédant alors une moyenne générale excellente, il est fort probable que le système le recommande aux utilisateurs recherchant des objets similaires.
- Attaques de type *nuke* : Le but est identique au type précédent : donner à l'objet ciblé une

note globale supérieure à ses concurrents. La manière de procéder est cependant différente. Les profils créés par les auteurs de l'attaque ne serviront plus à voter positivement, mais à détériorer les objets concurrents. Le résultat final est semblable aux attaques *push*, la note de l'objet à favoriser se trouve supérieure à celle de ses concurrents, et il a donc plus de chance d'apparaître dans les objets recommandés.

Il existe plusieurs modèles d'attaques basés sur ces deux idées. Ainsi, nous trouvons l'attaque aléatoire et l'attaque basée sur la moyenne. Dans le premier cas, les profils créés notent positivement l'objet ciblé tandis que les autres reçoivent une cote aléatoire. Concernant le modèle basé sur la moyenne, la méthode est similaire à l'exception des notes aléatoires remplacées par une valeur se situant autour de la moyenne, précédemment calculée. Cependant, notons que l'obtention de la moyenne générale peut, dans certains cas, s'avérer compliqué.

Il existe également un type d'attaque visant tout particulièrement les systèmes possédant un modèle *item-based*. L'objectif ici est de forcer la recommandation d'un objet particulier pour un groupe d'utilisateurs bien précis, dont les préférences sont connues. Pour cela, les personnes malicieuses insèrent un grand nombre de nouveaux utilisateurs, notant les objets de manière à augmenter de manière considérable le degré de similitude entre l'objet cible et ceux appréciés par le groupe. Possédant un degré de similitude élevé, il est dès lors fort probable que l'objet visé par les attaquants soit recommandé au groupe d'utilisateurs cibles [11].

2.4 Systèmes de recommandation contextualisés

2.4.1 Introduction

Les technologies de recommandation et les systèmes contextualisés sont aujourd'hui couramment utilisés. Néanmoins, ces deux techniques sont rarement rencontrées conjointement.

Deux chercheurs, Gediminas Adomavicius et Alexander Tuzhilin, ont pourtant réalisé une recherche complète à travers laquelle ils étudient les possibilités d'interaction entre les systèmes

de recommandation et les technologies de contextualisation [8]. De cette étude, présentée dans la suite de cette partie, ils tireront plusieurs résultats concluants.

Dans cette section, nous commencerons par détailler les motivations des auteurs à étudier les systèmes de recommandation contextualisés (CARS). Nous verrons ensuite l'objectif que de tels systèmes tendent à atteindre.

Ensuite, nous citerons et expliquerons les deux catégories principales de CARS, selon la manière dont les informations de contexte sont utilisées. Nous nous concentrerons sur l'une d'elles, *contextual preference elicitation and estimation*, et aborderons les trois paradigmes existants appartenant à cette approche.

2.4.2 Motivations

La plupart des approches des systèmes de recommandation se concentrent actuellement sur la recommandation des objets les plus pertinents pour des utilisateurs individuels, sans prendre en considération les informations contextuelles, telles que le temps, le lieu et l'entourage d'autres personnes. En d'autres termes, les systèmes de recommandation se contentent d'utiliser deux types de données, les utilisateurs et les objets, et ne les mettent pas en contexte lorsqu'ils prodiguent des recommandations.

Cependant, dans beaucoup de domaines, la prise encompte des seules dimensions d'utilisateurs et d'objets est insuffisante. Par exemple, un touriste ne recherchera pas toujours le même type d'attraction selon la période de l'année. Ainsi, un système de recommandation devra éviter de recommander des attractions en plein air nécessitant une température clémente (plage, piscine...) sous la neige.

2.4.3 Objectif

Traditionnellement, un moteur de recommandation utilise un ensemble de notes initiales, explicitement fournies ou inférées par le système. Une fois ces notes initiales spécifiées, le système essaie d'estimer la fonction de notation R

$$R : User \times Item \rightarrow Rating$$

pour les paires (utilisateur, objet) qui n'ont pas encore été notées par les utilisateurs. Lorsque la fonction R a été estimée pour tout l'espace $User \times Item$, un système de recommandation peut proposer pour chaque utilisateur l'objet ayant obtenu la meilleure note. De tels systèmes sont appelés traditionnels ou bidimensionnels (2D) puisqu'ils utilisent uniquement les dimensions $User$ et $Item$ dans le processus de recommandation.

Dans le cas des systèmes de recommandation contextualisés, les informations de contexte sont incorporées comme des catégories additionnelles de données dans le processus de recommandation. En d'autres termes, la nouvelle fonction de notation R est définie comme

$$R : User \times Item \times Context \rightarrow Rating$$

2.4.4 Les différentes approches

Les différentes approches pour utiliser les informations contextuelles dans les processus de recommandation peuvent être catégorisées en deux groupes :

- La recommandation par *context-driven querying and search*
- La recommandation par *contextual preference elicitation and estimation*

La recommandation par *context-driven querying and search* a été utilisée par une large variété de systèmes de recommandation touristiques et mobiles [43] [7] [20]. Les deux premiers systèmes sont présentés dans les sections 3.3 (p.35) et 3.2 (p.33) . Les systèmes suivant cette approche utilisent l'information contextuelle (obtenue soit directement à partir de l'utilisateur, par ex. son humeur ou ses intérêts, soit à partir de l'environnement, par ex. l'heure locale, la météo ou la localisation actuelle) pour interroger un certain catalogue de ressources (par ex. de restaurants) et présenter celles correspondant le mieux à l'utilisateur (par ex. les restaurants qui sont actuellement ouverts, à proximité de l'utilisateur, convenant à ses goûts...).

FIGURE *2.1 – Processus des systèmes de recommandation traditionnels. Source : [8]*

La recommandation par *contextual preference elicitation and estimation* constitue une tendance plus récente dans la littérature des systèmes de recommandation contextualisés [9] [35] [38] [48]. En contraste avec la première approche, les techniques de cette catégorie tentent de modéliser et d'apprendre les préférences d'un utilisateur, en enregistrant ses interactions et celles des autres utilisateurs avec le système ou en obtenant des *feedbacks* de préférence sur différents objets précédemment recommandés.

Pour modéliser les préférences contextuelles et générer les recommandations, ces techniques adoptent typiquement les méthodes de *collaborative filtering*, *content-based filtering* et hybrides. Ces méthodes ont été précédemment détaillées dans la partie 2.3.2 (p.13) de la section précédente. Dans certains cas, des techniques intelligentes d'analyse de données issues des domaines du *data mining* et du *machine learning* (comme les *Bayesian classifiers*) sont utilisées.

Les auteurs indiquent qu'il est également possible de concevoir des applications combinant les deux approches dans un seul système. Bien que les deux approches offrent des pistes de recherche intéressantes, Gediminas Adomavicius et Alexander Tuzhilin se concentrent sur la deuxième. Nous la détaillerons dans la section suivante.

2.4.5 L'approche par *contextual preference elicitation and estimation*

Comme expliqué précédemment (cf. 2.4.3 (p.22)), les systèmes de recommandation traditionnels peuvent être décrits comme une fonction prenant en entrée les préférences partielles d'un utilisateur et produisant une liste de recommandations pour chaque utilisateur en sortie. La figure 2.1 (p.23) illustre ce procédé.

Ces systèmes sont donc construits uniquement sur la base des préférences partielles des uti-

lisateurs. Les systèmes de recommandation contextualisés sont quant à eux construits sur base des préférences partielles contextualisées des utilisateurs. Les données ne sont donc plus représentées par des enregistrements de type $U \times I \times R$ mais possèdent une dimension supplémentaire et prennent la forme $U \times C \times I \times R$. Dès lors, chaque enregistrement spécifique ne se contente plus d'indiquer à quel point un utilisateur apprécie un objet spécifique, mais contient également l'information contextuelle permettant d'indiquer dans quelles conditions l'objet a été apprécié.

Dès lors, il est possible d'intégrer l'information à propos du contexte courant à plusieurs endroits du processus de recommandation. Plus spécifiquement, selon le composant utilisant le contexte, les systèmes de recommandation possédant une approche *contextual preference elicitation and estimation* peuvent prendre l'une des trois formes suivantes, illustrées à la figure 2.2 (p.25) :

- *Contextual pre-filtering* : Dans ce paradigme, l'information contextuelle mène la sélection ou la construction des données pour un contexte spécifique. En d'autres termes, l'information du contexte courant est utilisée pour sélectionner ou construire l'ensemble pertinent d'enregistrement de données (c.-à-d., les notations). Ensuite, les notations peuvent être prédites en utilisant tout système de recommandation bidimensionnel sur les données sélectionnées.
- *Contextual post-filtering* : Dans ce paradigme, l'information contextuelle est initialement ignorée, et les notations sont prédites en utilisant tout système de recommandation bidimensionnel sur l'entièreté des données. Ensuite, l'ensemble résultant des recommandations est ajusté (contextualisé) pour chaque utilisateur, en utilisant l'information contextuelle.
- *Contextual modeling* : Dans ce paradigme, l'information contextuelle est directement utilisée dans la technique de modélisation comme part intégrante de l'estimation des notations.

Si les paradigmes *Contextual pre-filtering* et *Contextual post-filtering* possèdent l'avantage de pouvoir utiliser les systèmes de recommandation bidimensionnels, ce n'est cependant pas le cas du troisième paradigme.

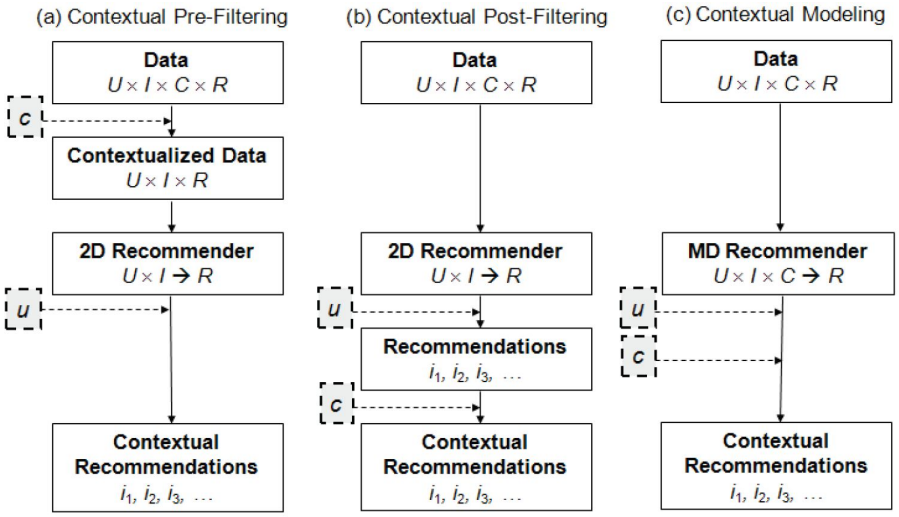

FIGURE *2.2 – Paradigmes pour l'incorporation du contexte dans les systèmes de recommandation. Source : [8]*

Le choix entre les deux premiers paradigmes dépend fortement de l'application. Néanmoins, les techniques de *post-filtering* possèdent une complexité plus élevée. En effet, dans le cas du *pre-filtering*, seul un sous-ensemble relativement réduit des données est utilisé pour l'estimation des notations.

2.5 Ontologies

2.5.1 Introduction

Actuellement, le contenu du web est conçu de manière à être parcouru par des êtres humains. Si les programmes informatiques peuvent en déduire la structure et en afficher le contenu de manière correcte, ils sont cependant incapables d'en connaître la signification. Ils ne disposent généralement pas de méthodes systématiques afin de traiter le contenu d'un document sur base de leur sémantique.

Ces dernières années, de nombreuses recherches dans le domaine du web sémantique ont été

effectuées afin de donner la possibilité aux programmes informatiques de traiter le contenu des documents sur la base de la sémantique. Cette capacité ouvrirait la voie à de nouveaux procédés d'automatisations et certaines tâches, aujourd'hui manuelles, pourraient être traitées de manière automatique [27].

Concrètement, le web sémantique va structurer le contenu sémantique des documents. Cette opération va ensuite permettre aux programmes de les interpréter et de raisonner sur leurs bases.

De nombreux langages ont été définis. Citons par exemple RDF/RDFS (*Resource Description Framework* et RDF Schema), DAML et OIL (DARPA *Agent Markup Language* et *Ontology Inference Layer*), et OWL (*Web Ontology Language*) qui constitue une extension de RDF/RDFS [28] [12].

2.5.2 Définition

Le terme ontologie est emprunté à la philosophie et signifie l'étude de l'être et de ses relations [2]. Ce terme a ensuite été repris dans le domaine de l'informatique dans les années 2000. Une ontologie y est l'ensemble structuré des termes et concepts représentant le sens d'un champ d'informations. Aujourd'hui, les ontologies sont considérées comme le pilier du web sémantique.

Leur objectif principal est de modéliser un ensemble de connaissances de manière structurée. Elles sont aujourd'hui utilisées dans de nombreux domaines tels que l'intelligence artificielle, le génie logiciel et l'informatique biomédicale.

2.5.3 Composants

Les ontologies partagent une structure commune [31] :

- Individus : Choses décrites par les ontologies. Cela peut consister aussi bien en des objets concrets tels des personnes, appareils... qu'en des objets abstraits tels le métier d'une personne.
- Classes : Les classes, aussi appelées types ou concepts, représentent les groupes d'individus qui partagent des caractéristiques communes. Certains langages (comme OWL) peuvent aussi définir une classe en fonction des individus y appartenant.

- Relations : Les relations décrivent la manière dont les individus interagissent. Ces relations peuvent être décrites de manière directe entre les individus (ex : Le travail x *est rédigé par* Roger). Mais elles peuvent aussi être définies entre des classes (ex : Un livre *est rédigé par* une personne).
- Attributs : Les attributs sont les différentes caractéristiques qui peuvent être attribuées à une classe.

Néanmoins, d'autres composants peuvent être intégrés selon l'objectif poursuivi dans la construction de l'ontologie.

2.5.4 Le langage de représentation OWL

Il est évident que les ontologies doivent posséder un langage de représentation. S'il en existe plusieurs, ceux-ci se rapprochent généralement des langages et formalismes logiques, tel celui des prédicats. L'utilisation de tels langages permet de spécifier des contraintes sémantiques en évitant au développeur de devoir les implémenter de manière explicite.

Dans ce domaine, OWL (Web Ontology Langage) s'est rapidement démarqué et est devenu en 2004 un standard W3C. Nous présentons ici, par le biais d'un exemple (cf. figure 2.3 (p.28)), son mode d'utilisation.

Par l'utilisation du langage OWL, toute arborescence se doit de posséder la classe « thing » comme racine. Toute classe créée par la suite est donc une sous-classe de « Thing », et chaque individu est un membre de cette superclasse.

Le langage OWL possède des relations prédéfinies. Outre la relation « isa », permettant de déclarer des sous-classes (et représentée par les flèches dans notre exemple), il est aussi possible de déclarer que deux classes sont équivalentes, « equivalentClass » (elles possèdent chacune les mêmes individus), ou inversement, les déclarer disjointes, « disjointWith » (deux classes ne peuvent avoir d'instances communes). Ce n'est par exemple pas le cas des classes *cheeseTopping* et *VegetableTopping*, toutes deux reliées à la classe *cheeseyVegetableTopping*.

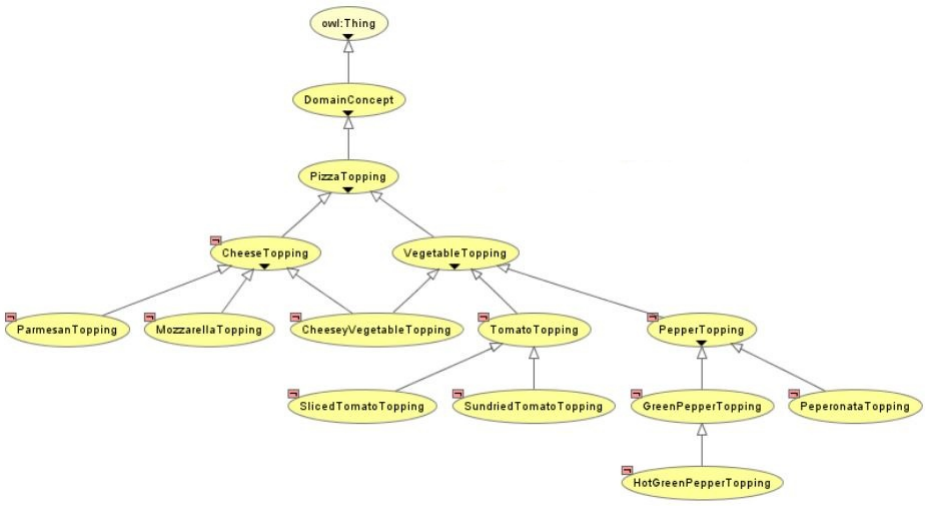

FIGURE *2.3 – Modèle d'ontologies de pizzas. Source : CO-ODE.org*

OWL possède également trois relations à destination des instances. « sameAs » permet de déclarer que deux instances sont identiques tandis que « differentFrom » indique l'opposé. Il est également possible, via la relation « allDifferent », de déclarer tous les individus d'une même classe mutuellement différents des uns des autres.

OWL offre aussi la possibilité de créer des propriétés, et fournit également un mécanisme d'héritage. Il existe deux catégories de propriétés : celles reliant une instance à une autre (propriété d'objet), et celles associant des valeurs à une instance (propriété de données). Concrètement, une propriété est définie par trois éléments : le domaine (classes pour lesquelles la propriété est utilisée), la relation et l'image (classes reliées au domaine par la propriété).

2.5.5 Les moteurs d'inférences

Selon la définition, une inférence est « un processus de raisonnement qui s'appuie sur des connaissances acquises, et qui s'articule autour de règles fondamentales pour permettre d'obtenir de nouvelles informations ». [1]

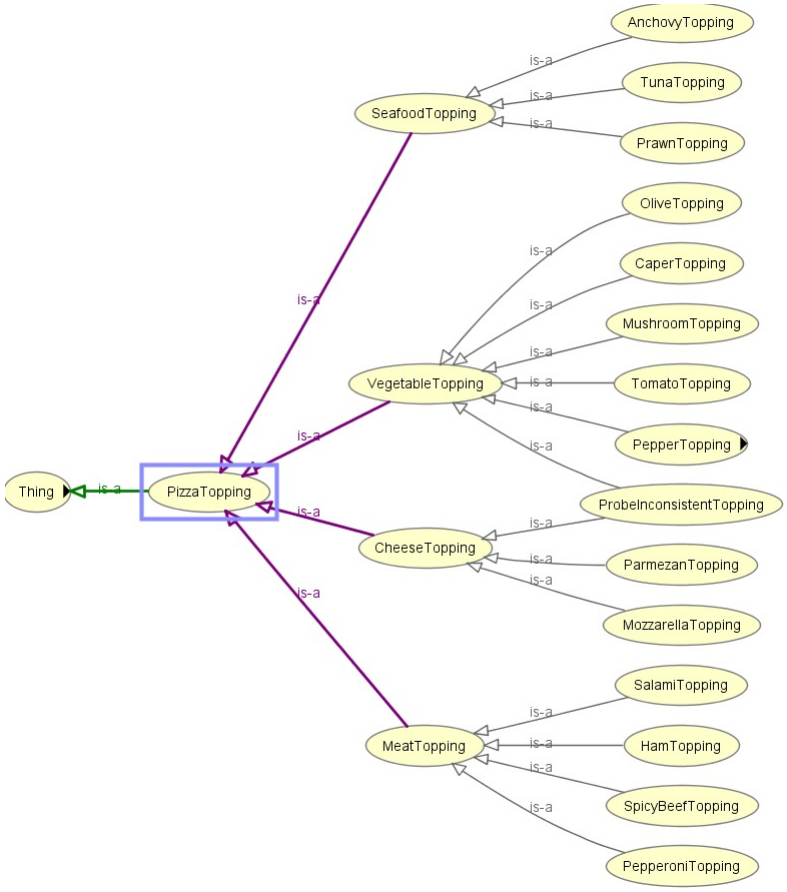

FIGURE *2.4 – Garnitures de pizzas. Source : CO-ODE.org*

Les moteurs d'inférences permettent aux systèmes experts de procéder à des raisonnements logiques et de déduire de l'information sur une base de règles et de connaissances. Ils obtiennent de nouvelles informations en interprétant les ontologies. Donnons un exemple (cf. 2.4 (p.29)). Chaque garniture de pizza possède une classe, et est une sous-classe d'un type d'aliment (par exemple, *oliveToping* est une sous-classe de la classe *vegetableTopping*). Il est dès lors possible

de déduire si une pizza est végétarienne par une propriété du type :

$$Pizzaand(hasToppingonlyVegetableTopping)$$

Selon cette règle, le moteur d'inférence vérifiera que toutes les garnitures appartiennent à la classe *vegetableTopping*, illustrée à la figure 2.4 (p.29) . En associant cette règle à une classe (*vegetarianPizza* par exemple), ses individus pourront alors être inférés de manière automatique.

Il existe actuellement plusieurs moteurs d'inférences d'ontologies tels que Racer, Pellet, Fact, Fact++, Surnia, F-OWL et Howlet. La plupart sont conçus pour raisonner sur les logiques de description, mais acceptent en entrée des fichiers OWL. Certains moteurs d'inférence ne peuvent raisonner qu'au niveau terminologique (c'est-à-dire au niveau des concepts et des propriétés) alors que des moteurs comme Pellet et Racer permettent de raisonner aussi sur les instances de concepts [26].

2.5.6 Les ontologies comme support du contexte

Au début des années 2000, les chercheurs ont commencé à utiliser les technologies sémantiques pour modéliser le contexte. Durant les neuf dernières années, les ontologies ont été fréquemment utilisées dans la conception des systèmes contextualisés. Cependant, les recherches étaient plus motivées par le besoin de trouver un moyen efficace de modéliser le contexte que par sa conceptualisation à proprement parler. En effet, les auteurs de cette décennie n'ont donné aucune nouvelle définition du contexte, et ont réutilisé celle proposée par Dey comme présentée à la section 2.2 (p.10) .

Par la suite, une analyse exhaustive des méthodes de gestion du contexte, réalisée par Strang et Linnhoff-Popien en 2004 [42], a indiqué que les ontologies sont l'outil le plus adéquat pour la gestion des informations de contexte.

Pour cela, les auteurs ont établi une liste de critères sur lesquels les différentes méthodes ont

été évaluées :

- Composition distribuée (dc) : chaque système mobile contextualisé est intrinsèquement un système distribué. C'est pourquoi la technique de modélisation du contexte doit être adaptée à cette caractéristique.
- Validation partielle (pv) : il doit être possible de valider partiellement les connaissances de contexte. Aussi bien la structure que les instances doivent pouvoir être validées contre le modèle de contexte en utilisation.
- Richesse et qualité de l'information (qua) : Tout comme la qualité d'une information peut varier entre différents censeurs, la richesse de l'information est influencée par la dégradation d'un même censeur à travers le temps. Dès lors, un modèle de contexte approprié pour un usage mobile contextualisé doit supporter une indication de la qualité et de la richesse des informations.
- Incomplétude et ambiguïté (inc) : Les informations recueillies peuvent dans certains cas être incomplètes et/ou ambigües. Cela doit être pris en compte par le modèle, par exemple par interpolation des données incomplètes au niveau des instances.
- Niveau de formalité (for) : Il est important de pouvoir décrire les faits contextuels et les relations d'une manière précise. Cela implique que les différentes parties d'un système mobile contextualisé partagent les mêmes interprétations et les mêmes définitions à propos des termes utilisés.
- Application aux environnements existants (app) : Du point de vue de l'implémentation, il est important qu'un modèle de contexte soit applicable au sein des infrastructures existantes.

Différentes techniques de validation ont été évaluées par les auteurs sur ces différents critères. Les résultats de cette analyse sont présentés dans le tableau 2.1 (p.32) . Il ressort de cette analyse que la méthode ayant le mieux répondu aux différents critères est celle qui se base sur les ontologies.

Cette recherche a eu un impact décisif sur les systèmes contextualisés, puisque dès lors l'écrasante majorité d'entre eux se fieront à cette étude, et choisiront les ontologies pour modéliser

Approach - Requirem.	dc	pv	qua	inc	for	app
Key-Value Models	/	/	-	-	-	+
Markup Scheme Mod.	+	++	/	/	+	++
Graphical Models	-	/	+	/	+	+
Object Oriented Mod.	++	/	/	/	++	-
Logic Based Models	++	/	/	./	++	-
Ontology Based Mod.	++	++	+	+	++	+

TABLE *2.1 – Indicateurs d'adéquation - Source : Strang et Linnhoff-Popien [42]*

et utiliser leurs données contextuelles. C'est ainsi le cas de presque tous les systèmes présentés dans la partie 3 (p.33) .

Cette avancée a ouvert la voie à des systèmes permettant le partage et la réutilisation de contexte. De plus, l'utilisation des technologies sémantiques permet non seulement de valider la consistance du modèle de contexte, mais aussi d'inférer de nouvelles connaissances implicites grâce au raisonnement sur les ontologies.

Aujourd'hui, plusieurs ontologies de contextes ont été développées. Les plus connues sont SOUPA et CONON, dont le modèle est présenté par la figure 3.3 (p.42) dans la section 3.5 (p.41) .

Chapitre 3

État de l'Art

3.1 Introduction

Au cours des 10 dernières années, un nombre croissant de recherches ont eu pour sujet les applications mobiles *context-aware*. Ce chapitre a pour objectif d'en présenter quelques-unes, en pointant leurs avantages et leurs faiblesses. Ainsi, nous détaillerons les projets Cyberguide, COMPASS, Flame08, CONON, SOUPA, Friend of a Friend, les travaux de Chien-chih Yu, CONCERT et CAMTON.

Nous terminerons ce chapitre par une synthèse mettant en avant les défauts les plus récurrents, à partir desquels nous proposerons une nouvelle architecture dans le chapitre 5 (p.67) .

3.2 Cyberguide

Dans le domaine du tourisme, Cyberguide [7] fut l'une des premières initiatives en 1997. D'après ses auteurs, Cyberguide avait pour but « d'être une application capable de connaître la position du touriste, ses intentions, qui peut prédire et répondre aux questions qu'il pourrait se poser, et qui donne la possibilité d'interagir avec d'autres personnes et l'environnement ».

Pour y arriver, les développeurs envisageaient l'utilisation de la réalité augmentée. L'appareil mobile pourrait être équipé d'une caméra. À partir des images filmées, l'application pourrait détecter la position de l'utilisateur.

Afin d'expliquer concrètement les objectifs de leur application, les développeurs comparent Cyberguide avec les audioguides souvent disponibles dans les musées. Ceux-ci ne fonctionnent généralement que dans les musées pour lesquels ils ont été conçus. Le but recherché par Cyberguide serait la création d'un audioguide universel, capable de fournir des informations au touriste, quelle que soit sa position.

De plus, les auteurs souhaitaient y intégrer un système de création d'itinéraires selon des points d'intérêts préalablement sélectionnés par l'utilisateur.

Concernant l'architecture, les concepteurs voyaient Cyberguide comme un ensemble de services pouvant proposer à l'utilisateur une expérience touristique complète. Les différents composants présents dans l'application sont :

- Composant cartographie
- Composant information
- Composant de position
- Composant de communication

L'objectif d'une telle approche est d'obtenir une application modulaire et extensible, rendant possible l'ajout de nouveaux composants (et donc de nouveaux services) dans le futur.

Néanmoins, l'une des principales faiblesses de Cyberguide est la nécessité de disposer de capteurs intégrés à l'environnement afin de pouvoir localiser les appareils mobiles dans les bâtiments.

En effet, à l'époque, les auteurs se sont heurtés à de nombreuses limitations technologiques (connection limitée, mauvaise réception GPS...) qui ne sont aujourd'hui plus d'actualité. Malgré

ces difficultés, un prototype fonctionnant sur Apple Messenger a pu être conçu.

Une autre limitation de Cyberguide est liée à son architecture. Celle-ci prévoit en effet de placer toute l'information touristique au sein de l'appareil mobile. C'est pourquoi Cyberguide était intrinsèquement limité à un fonctionnement dans des espaces relativement restreints, sous peine de saturer l'espace mémoire de l'appareil.

En raison de ses fonctionnalités, Cyberguide est devenu un pionnier des applications touristiques contextualisées et a ouvert la porte à d'autres projets tels que COMPASS ou FLAME08 en 2004.

3.3 COMPASS

L'application COMPASS, *COntext-aware Mobile Personal ASSistant*, est une application *context-aware* à destination des touristes, intégrant un moteur de recommandation [43]. Développé par Mark van Setten, Stanislav Pokraev et Johan Koolwaaij, son but est de proposer des points d'intérêts en fonction du thème choisi par l'utilisateur (paysages, architecture, hôtel...) et de son profil. La sélection de ces intérêts se fait en deux parties : dans un premier temps, seuls les points d'intérêts correspondant aux critères définis comme *hard criterions* sont retenus. Il s'agit par exemple de la localisation : les points d'intérêts se trouvant à plus d'une certaine distance ne figurent pas dans l'ensemble des résultats. Une note est ensuite attribuée à chacun des PoI (points d'intérêts) retenus. Celle-ci est attribuée selon la manière dont ils répondent à différents critères, appelés *soft criterions*.

Les points d'intérêts retenus sont ensuite affichés sur une carte et le touriste peut obtenir des informations les concernant. De plus, COMPASS veut être capable d'intégrer des *web services* proposés par des tiers. Dès lors, des services peuvent être associés à certains résultats. Il serait par exemple possible de réserver une chambre d'un hôtel à partir de l'application.

L'architecture de COMPASS est composée de 4 groupes : les services, la plate-forme WASP, l'application COMPASS et le système de recommandation.

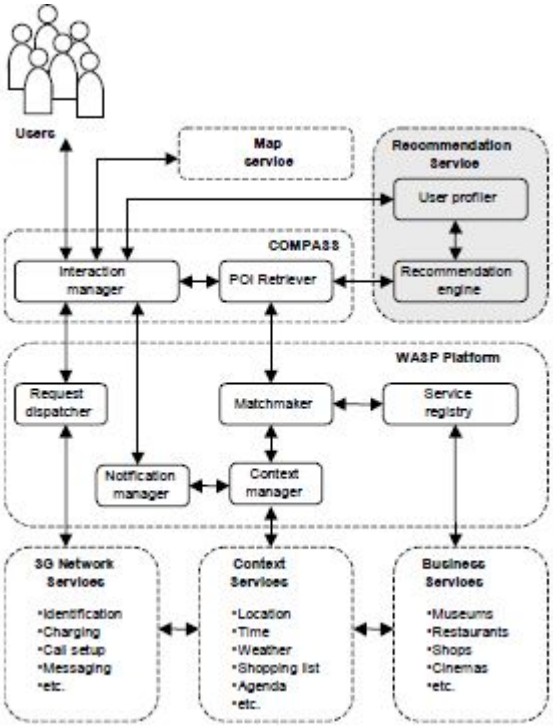

FIGURE *3.1 – Architecture de COMPASS. Source : [43]*

Le premier rassemble les différents services utilisés par l'application. On y trouve les services de communication (fournis par les opérateurs mobiles), les services permettant d'obtenir le contexte de l'utilisateur (localisation, heure, agenda...), et les services détenant les points d'intérêts et leurs informations.

La plateforme WASP est connectée aux différents services et récolte le contexte des utilisateurs ainsi que leurs requêtes. En fonction de celles-ci, elle rassemblera les points d'intérêts via certains services correspondant au thème demandé.

De plus, cette plateforme permet de gérer les connexions mobiles de l'utilisateur. Celui-ci peut donc passer d'un opérateur réseau à un autre de manière transparente.

Constituée d'un *Notification Manager*, la plateforme WASP vérifie régulièrement le contexte de l'utilisateur (son profil, sa localisation par exemple) et le met à jour si nécessaire. Ces changements seront directement pris en compte pour les requêtes futures du touriste.

Les informations reçues par le *Notification Manager* proviennent du *Context Manager*. Celui-ci se charge d'obtenir des informations à partir des différents services connectés. Il est aussi capable d'inférer de nouvelles données (détecter si l'utilisateur est à pied ou en voiture par exemple).

Enfin, le *Matchmaker* permet de sélectionner les services correspondant aux requêtes de l'utilisateur. Pour ce faire, celui-ci est relié au registre contenant les descriptions des services disponibles et sur base du profil de l'utilisateur et moyennant les ontologies, retire ceux qui ne correspondent pas aux critères.

L'application COMPASS est constituée de deux composants. Le premier, l'*Interaction Manager*, a pour but d'aider l'utilisateur à créer sa requête. Cela se fait grâce à une interface graphique qui affiche spontanément une liste de requêtes possibles lorsque l'utilisateur navigue à travers son application mobile.

Lorsque la requête a été rédigée, celle-ci est transférée au *POI Retriever*, deuxième composant de l'application COMPASS. Celui-ci, en communiquant avec le *Matchmaker* de la plateforme WASP, est chargé de récupérer les intérêts potentiellement pertinents pour le touriste.

Cette liste est ensuite envoyée au système de recommandation qui assigne un score pour chaque intérêt.

Le système de recommandation utilisé par COMPASS utilise plusieurs stratégies différentes (*filtering*, *case-based reasoning* (CBR), *item-item filtering* et *category learning*) afin d'obtenir les résultats les plus pertinents possible. Pour ce faire, le système utilise des classes reliées entre elles selon un arbre. La classe « intérêt » est la classe *root* qui contient donc des sous-classes (correspondant aux catégories d'intérêts), et ainsi de suite. Pour chaque classe, une stratégie de prédiction peut être associée. Lors du calcul, le système utilisera la stratégie associée à la classe correspondant aux points d'intérêts. Si aucune stratégie n'a été associée à la classe utilisée, celle de la première superclasse en possédant une sera utilisée. Une stratégie par défaut est évidemment associée à la classe *root* « intérêt ».

De plus, les développeurs de COMPASS ont implémenté des techniques de prédiction afin de prendre en compte les facteurs contextuels de l'utilisateur. L'une d'entre elles consiste notamment à pondérer le score d'un point d'intérêt en fonction du temps écoulé depuis la visite d'un autre PoI similaire par l'utilisateur. Le profil de celui-ci est aussi pris en compte. De cette manière, les scores donnés par le système de recommandation dépendent du profil de chaque utilisateur, de ses préférences et de ses *feedbacks*.

Enfin, notons que du point de vue de l'architecture, le système de recommandation ne fait pas partie de l'application COMPASS. La raison avancée par les créateurs est de pouvoir le rendre disponible à d'autres applications touristiques que COMPASS.

Pour tester leur application, les développeurs ont distribué un prototype à des utilisateurs en leur demandant de remplir un sondage. Ils ont pu en tirer que la raison pour laquelle les utilisateurs n'avaient pas apprécié l'utilisation de COMPASS était souvent la même : le système était devenu trop intelligent, trop décideur et donnait un sentiment de soumission au touriste.

C'est en effet un problème récurrent des systèmes de recommandation : en limitant le nombre de résultats, l'utilisateur a l'impression de ne pas obtenir toute l'information. Concernant le domaine du tourisme, l'utilisateur peut avoir le sentiment de passer à côté d'attractions pouvant

l'intéresser. Notre système tentera de limiter ce problème en proposant un maximum de points d'intérêts. D'autres mécanismes (l'utilisateur pourra réordonner la liste des résultats à sa guise...) seront également mis en place afin de laisser un sentiment de liberté à l'utilisateur.

3.4 FLAME08

En 2004 fut lancé le projet FLAME08 [45], à destination des Jeux Olympiques de Pékin de 2008. Celui-ci consistait à développer une plate-forme d'intégration de *web services*. Ainsi, les touristes pouvaient être guidés par le biais d'une application mobile à travers la ville et les différents stades selon le programme des jeux. Pour cela, l'application pouvait proposer des points d'intérêts aussi bien de manière spontanée que sur une demande explicite de l'utilisateur, toujours selon son profil et sa situation actuelle.

L'architecture de FLAME08, constituée de 3 composants principaux, est montrée sur la figure 3.2 (p.40) . Le fonctionnement du *framework* se déroule comme suit : un profil est associé à chaque *web service* connecté. Ce profil est constitué d'un ensemble d'attributs décrivant au mieux le service. Parallèlement, un procédé similaire est opéré sur les utilisateurs du *framework* dont les caractéristiques sont détectées par des senseurs (GPS, horloge...) ou peuvent être déduites grâce à l'utilisation d'un moteur d'inférence.

Ensuite, une correspondance est effectuée entre le profil mis à jour du visiteur et ceux des services disponibles. S'il y a correspondance, FLAME08 proposera le service sur le smartphone du touriste.

La pertinence des résultats dépend donc fortement de la qualité des profils générés, aussi bien pour les utilisateurs que pour les services liés. Pour leurs créations, les développeurs ont dès lors décidé d'utiliser les ontologies.

Concrètement, chaque profil est constitué de plusieurs parties. Premièrement, il est constitué

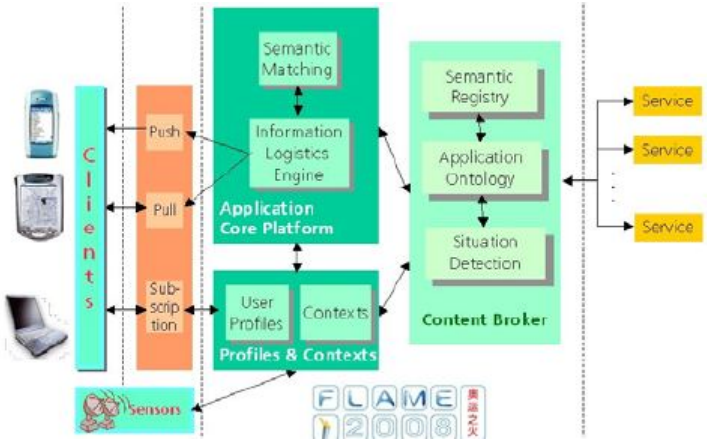

FIGURE *3.2 – Architecture de FLAME08. Source : [45]*

de l'information dite statique, généralement directement fournie par l'utilisateur et qui doit être maintenue à jour par celui-ci. Il s'agit par exemple de ses préférences, son agenda... Ensuite, nous avons les informations de situation et de contexte. Celle-ci est obtenue par le biais de senseurs, disposés sur le téléphone du touriste (pour obtenir sa localisation...), ou par l'intermédiaire d'API's (température, heure...).

À partir de ces informations et des relations entre elles, il est souvent possible d'inférer de nouvelles données (la vitesse de déplacement, déduite depuis l'évolution du temps et de la localisation, pourrait indiquer la manière dont le visiteur se déplace à travers la ville) et donc d'enrichir son profil. Ces informations seront précieuses lors du *matching*.

Celui-ci est réalisé dès que le contexte de l'utilisateur change de manière significative, ce qui sera interprété par le moteur d'inférence comme un changement de situation. Pour les auteurs, une situation est un contexte de haut niveau, inféré à partir de plusieurs informations de contexte de plus bas niveau. Par exemple, un utilisateur présent dans le stade d'athlétisme à l'heure de la finale du 100m sera considéré comme spectateur de cet évènement. De nouveaux *web services* adaptés à cette nouvelle situation lui seront alors proposés.

FLAME08 analyse donc le contexte de l'utilisateur de manière continue. Outre le fait que cela peut être la cause de problèmes concernant le respect de la vie privée, une application fonctionnant en permanence diminue de manière drastique l'autonomie du téléphone. Or, aujourd'hui encore, l'autonomie reste l'un des points faibles majeurs concernant les smartphones...

3.5 L'ontologie CONON

Dans la même année, Xiao Hang Wang et son équipe ont étudié l'utilisation des ontologies pour le développement d'un modèle de contexte, CONON (*CONtext ONtology*) [44]. Pour les auteurs, l'utilisation des ontologies est le meilleur procédé pour représenter de manière optimale les concepts du contexte, les partages de connaissances et le raisonnement sur celles-ci dans de larges environnements.

En effet, les ontologies permettent d'uniformiser et de fournir aux applications un ensemble de concepts communs. Par conséquent, le partage de connaissances s'en trouve grandement facilité. De plus, grâce à cette représentation, il est possible à partir de connaissances « de haut niveau » d'inférer de nouvelles données de plus bas niveau. Cet aspect a déjà été abordé dans la description du projet FLAME 08.

Lors de la création de ce projet, le domaine des systèmes *context-aware* était en pleine évolution, et ne disposait pas de standards reconnus. L'objectif de CONON était de pallier ce manque, de fournir un squelette de concepts que tout système utiliserait. Si CONON modélise ces entités appelées « de haut niveau », celui-ci fournit également des extensions flexibles contenant des concepts appartenant à certains domaines spécifiques. Une partie du modèle est montrée sur la figure 3.3 (p.42) .

Chaque concept est donc représenté par une entité possédant des attributs. Cependant, afin de pouvoir faire du raisonnement, il est nécessaire d'y associer des relations. Certaines relations sont définies par défaut dans le langage de représentation des connaissances OWL (*Ontology Web*

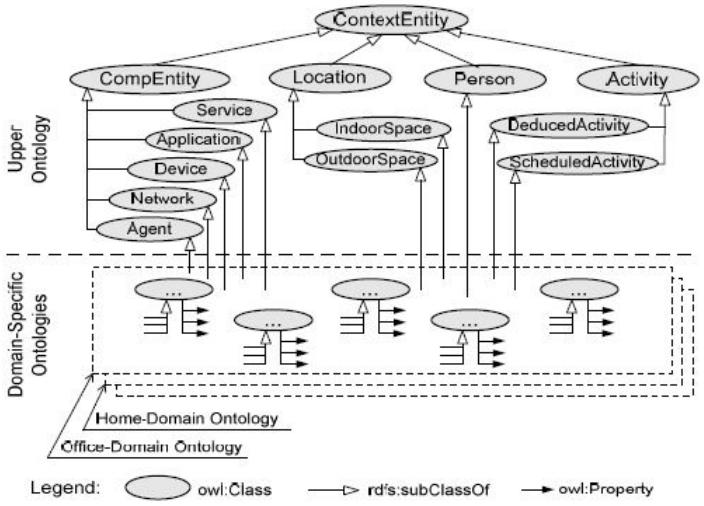

FIGURE *3.3 – Modèle partiel de Conon. Source : [44]*

Language), les autres sont créées en fonction des applications.

L'inférence se fait par l'utilisation de la logique du premier ordre. À titre d'exemple, on pourrait écrire : (*Tom* locatedIn *Home*) $\wedge$ (*Time* isTime *23*) => (*Tom* state *sleeping*).

Si cet exemple peut paraître anodin pour le lecteur, il démontre néanmoins la puissance que peuvent apporter les ontologies dans les systèmes de contextualisation. D'où l'importance pour les auteurs de procurer un modèle générique permettant une réutilisation des concepts.

Malheureusement, l'utilisation d'ontologies comporte aussi des inconvénients. Un prototype a été développé par l'équipe travaillant sur le projet CONON afin d'en mesurer les performances. Celui-ci a été construit en utilisant le Jena2 Semantic Web Toolkit et a été testé sur une station Linux équipée de 512MB de mémoire RAM et des processeurs : P3/600 MHz, P3/1.2 GHz, P4/2.4 GHz. L'objectif de cette phase de test était de pouvoir apprécier le comportement du raisonneur avec une large quantité de données.

Le résultat indique que le temps d'exécution évolue de manière exponentielle tandis que le nombre d'entités grandit de manière linéaire. Néanmoins, les auteurs arriveront à la conclusion suivante : le raisonnement est généralement faisable pour les applications n'étant pas *time-critical*, et à condition qu'il soit opéré sur des machines dédiées, celles des clients se contentant de recevoir les résultats.

3.6 L'ontologie SOUPA

SOUPA [21], *Standard Ontology for Ubiquitous and Pervasive Applications*, a été développée par les auteurs ayant précédemment mis au point CoBrA-ONT, une autre ontologie de contexte. SOUPA est cependant plus complète dans la mesure où l'attention a particulièrement été portée sur sa réutilisabilité, l'un des défauts de CoBra-ONT.

SOUPA est constituée de deux parties, présentées à la figure 3.4 (p.44) : SOUPA *Core*, constituée des ontologies génériques servant de base commune pour les systèmes contextualisés, et SOUPA *Extensions*, contenant les ontologies spécifiques au domaine d'utilisation.

Ainsi, l'entité *Person* fournit les concepts pour les informations de contact et les profils des personnes. Pour cela, elle contient des attributs tels que le nom, la date de naissance, le numéro de téléphone, l'adresse électronique ainsi que les relations sociales et professionnelles.

L'entité *Policy* fournit quant à elle les concepts de sécurité et de confidentialité applicables à des actions, représentées par l'entité *Action*, effectuées dans le système. Une instance de *Policy* permettrait d'autoriser ou de refuser certaines opérations, représentées par une instance d'*Action*.

Les entités *Agent* permettent la description d'agents, étant soit des utilisateurs humains, soit des entités informatiques. Les technologies orientées Agent utilisent trois ordres d'idées contenues dans l'entité *BDI* : croyances (*Facts*), objectifs (*Desires*) et intentions (*Intentions*).

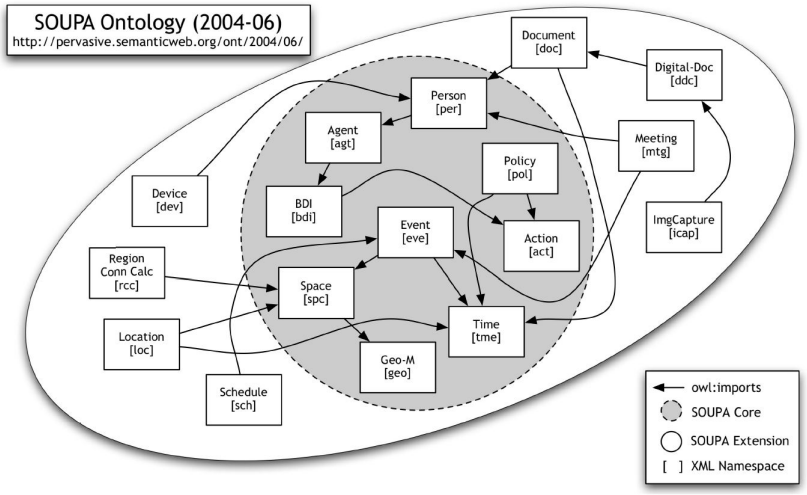

FIGURE *3.4 – Modèle de SOUPA. Source : [44]*

Les croyances représentent les faits connus par l'agent, mais non nécessairement vrais. Les objectifs représentent les buts tandis que les intentions représentent l'ensemble des plans mis au point par l'agent afin de les atteindre. Une intention peut donc être vue comme une action spécialisée issue de l'entité *Action* de SOUPA, qui possède en plus une précondition et un effet.

L'entité *Time* de SOUPA a été développée sur la base de la DAML *Time ontology*. Elle fournit les concepts pour exprimer le temps et les relations temporelles. Les concepts principaux sont *TemporalThing*, représentant tout ce qui est temporel et contenant *TimeInstant* (un point dans le temps) et *TimeInterval* (une période de temps). *TimeInstant* possède plusieurs relations afin de représenter l'ordre de deux points de temps. On y trouve par exemple *sameTimeAs*, *before* et *after*. *TimeIntervals* possède des relations du type *startsSoonerThan* et *startsLaterThan*.

Les ontologies *Space* et *Geo Measurement* fournissent les concepts pour représenter les régions géographiques, coordonnées spatiales et leurs relations ainsi que les conversions correspondantes. Le concept central est *SpatialThing*, utilisé pour représenter tout ce qui possède des coordonnées spatiales. Le rôle de *Geo Measurement* est de fournir du vocabulaire géographique typique, tel

que la longitude, la latitude, l'altitude...

L'ontologie *Event* est chargée de décrire les évènements possédant à la fois des propriétés temporelles et spatiales. Le concept *Event* décrit donc les activités et les calendriers et est fusionné avec *SpatialThing* et *TemporalThing*, résultant des concepts présentés précédemment.

Les ontologies d'extension de SOUPA ont été construites afin de supporter des scénarios d'applications spécifiques. Par exemple, on peut y trouver les ontologies *Meeting and Schedule*, permettant d'instancier les concepts des relations des meetings (programme, participants...)

3.7 Friend of a friend

Durant l'année 2006, Luca Buriano présente l'ontologie FOAF (*Friend of a Friend*) à destination des guides mobiles touristiques *context-aware* [18]. Si chaque touriste possède un profil, FOAF se concentre essentiellement sur les relations existantes entre ces personnes. En effet, grâce à sa propriété *knows*, FOAF est capable de construire un réseau social, un graphe où chaque personne, représentée par un noeud, est directement connectée à ses connaissances.

La formation de tels graphes permet la détection de groupes, utilisables par le moteur d'inférence de l'application. En effet, le tourisme est une activité qui se fait rarement en solitaire. Les touristes sont généralement en groupe, et les types d'activités, les préférences peuvent varier en fonction de la constitution de ce groupe. À titre d'exemple, il parait évident qu'un jeune touriste sera à la recherche de points d'intérêts différents selon qu'il est accompagné de sa famille ou de ses amis.

Détecter les groupes, trouver des sous-groupes, des groupes similaires... sont quelques-unes des représentations de haut niveau intéressantes qui peuvent être construites avec cette ontologie. Pour les obtenir, l'auteur énonce quelques possibilités telles que le *data mining*, le raisonnement statistique, les algorithmes de correspondances de graphes...

Lorsqu'une application touristique obtient de telles informations du contexte social de l'utilisateur, elle peut les utiliser comme une dimension supplémentaire pour ses systèmes de recommandation. Les fonctionnalités du guide touristique s'en trouveraient améliorées. On peut par exemple imaginer qu'un utilisateur visitant une région reçoive des points d'intérêts fortement différents selon les personnes qui l'accompagnent et leurs intérêts propres.

3.8 Chien-chih Yu et Hsiao-ping Chang

Par la suite, le développement des smartphones ne cesse de croître, donnant naissance à de plus en plus de recherches dans le domaine des applications mobiles personnalisées. Cependant, il existe encore un manque cruel de standards et les fonctionnalités des systèmes existants restent primitives. L'équipe de Chien-chih Yu et Hsiao-ping Chang l'a bien compris, et tente d'apporter une modélisation rigoureuse concernant le développement de l'architecture et du design de telles applications [47].

Ceux-ci commencent par relever les différents aspects qui se doivent d'être présents dans de telles applications. Ainsi, il est nécessaire de retrouver le profil utilisateur, une liste de points d'intérêts selon sa localisation et sa destination... Un exemple d'architecture est présenté à la figure 3.5 (p.47) .

D'après les auteurs, le touriste devrait avoir la possibilité de choisir des points d'intérêts de types différents (d'où la présence de plusieurs systèmes de recommandations dans le design présenté), et d'indiquer la durée souhaitée pour chaque activité. L'utilisateur serait donc capable de créer de manière précise son agenda touristique, qui pourrait aussi être affiché grâce à un service de cartographie, tel que Google Maps.

Il devrait être également possible d'optimiser la recommandation selon le contexte, tel que l'heure à laquelle la recherche est effectuée. Par exemple, il est fort probable qu'un visiteur

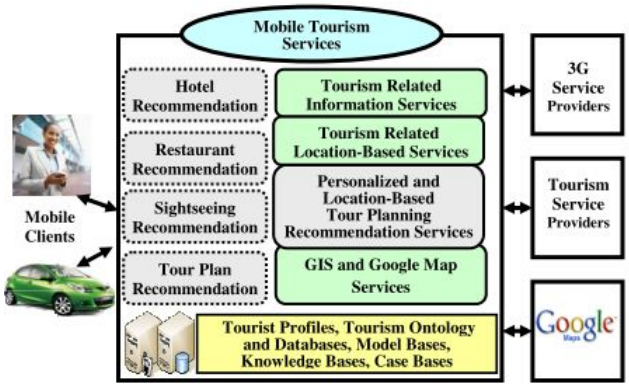

FIGURE *3.5 – Design d'un système de recommandation de guidage. Source : [47]*

envoyant sa requête aux alentours des heures de repas soit à la recherche d'un restaurant. L'application doit en être consciente.

Afin de mieux comprendre le fonctionnement d'un des systèmes de recommandation du système, l'équipe de développeurs illustre un cas précis d'utilisation, à savoir la sélection d'un hôtel. Pour cela, une base de données contenant les caractéristiques de chaque bâtiment est mise à disposition. Il est dès lors aisé de connaître le prix d'une nuit, les types et le nombre de chambres disponibles, la classe de l'hôtel, la présence d'un restaurant... Pour chaque critère, le système évalue les différents hôtels de la région et un classement général est ensuite généré. Le premier de la liste est proposé au touriste.

L'application est capable de générer un agenda pour la journée. Concrètement, celle-ci construit une liste de 7 activités selon un algorithme présenté à la figure 3.6 (p.48) . Pour cela, l'algorithme à besoin de la durée approximative de chaque activité. Lorsque le temps courant se situe entre midi et 14 heures (ou 18h et 21h), le système recommandera des restaurants se trouvant à proximité.

Si l'idée de création d'un agenda nous paraît intéressante, nous apprécions plus modérément le fait de forcer le type d'activité recommandé autour des heures de repas. En effet, les touristes

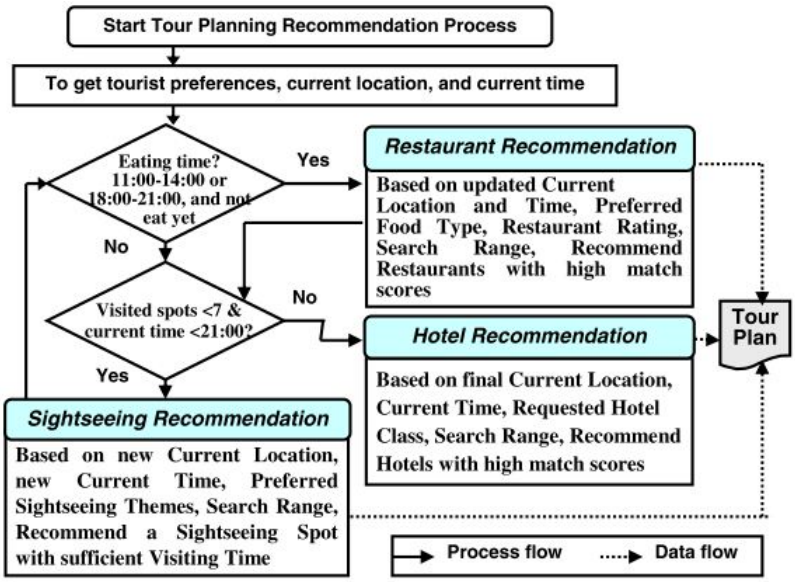

FIGURE *3.6 – Algorithme de création d'un agenda. Source : [47]*

peuvent manger à des heures décalées, en chemin entre deux attractions... Le système risque donc de rencontrer les mêmes écueils que le projet COMPASS. Ces systèmes sont « trop décideurs » aux yeux des utilisateurs. C'est pourquoi nous reprendrons l'idée de création d'agendas, mais notre système laissera la main à l'utilisateur concernant le type d'activités recommandées.

À la date de l'écriture de l'article (2009), les auteurs sont en train de développer un prototype. Celui-ci est constitué d'un serveur équipé de Windows XP, IIS Web Server 5.1, .NET Framework 2.0, Microsoft SQL Server 2005, ASP.NET 2.0, Web Services et Google Map API 2.0. L'application mobile proprement dite tourne sur l'émulateur Mobile 5.0 Smart Phone Emulator.

Les différentes captures d'écran proposées, visibles à la figure 3.7 (p.49) , illustrent avec précision les fonctionnalités présentes. L'utilisateur commence par sélectionner les types de services sur lesquels il souhaite effectuer la recherche. Durant la seconde étape, l'auteur précise ses différents critères de sélection, qui seront utilisés par le système de recommandation. Lorsque le système aura renvoyé le résultat contenant les points d'intérêts correspondant le mieux à la

FIGURE 3.7 – *Captures d'écran du prototype. Source : [47]*

demande du touriste, le téléphone affichera les informations détaillées ainsi qu'une route passant par ceux-ci, par le biais du service de cartographie Google Maps. Ces étapes sont visibles sur la 3e et la 4e image de la figure 3.7 (p.49) .

3.9 CONCERT

En 2010, Lamsfus et coll. ont développé CONCERT, un système de recommandation *rule-based* construit au-dessus d'un réseau d'ontologies (collection d'ontologies reliées par des propriétés) modélisant le contexte [29][30]. CONCERT récupère les informations nécessaires grâce à internet et aux senseurs intégrés à l'appareil de l'utilisateur uniquement. Il n'exige donc plus d'autres infrastructures externes comme c'était le cas avec Cyberguide dans les années 2000.

Le réseau d'ontologies utilisé, nommé Contology, a été développé pour l'occasion. Par rapport aux ontologies traditionnelles, les réseaux d'ontologies présentent les avantages de la flexibilité et de la modularité. Au moment de la publication de leur article, Contology était composé de 11 ontologies (visiteur, préférences, rôle, activité, environnement, appareil, réseau, motivation, activité, temps et services touristiques).

Afin de limiter les coûts liés aux connexions mobiles, Lamsfus et coll. ont utilisé une technologie de communication encore méconnue en Belgique : la radio numérique. Celle-ci, en plus de

faire office de support à des données audio, à l'instar de la radio FM classique, peut aussi servir à diffuser des images ou d'autres types d'informations. S'il n'est pas nécessaire de disposer d'une connexion internet pour récupérer les informations, le téléphone doit néanmoins être muni d'un décodeur spécifique.

À ce stade, nous observons un problème concernant la portée universelle de la solution CONCERT. En effet, l'utilisation de la radiodiffusion implique une restriction du champ d'action du système. Pour fonctionner, il est nécessaire qu'une antenne relais soit située à proximité et qu'un organisme ait mis en place un système CONCERT. Ce système se limite donc aux villes équipées spécifiquement et ne correspond pas aux exigences de notre système concernant l'universalité de l'application 6.3.2 (p.86) .

Les auteurs ont ainsi utilisé Journaline, un service de données standardisé ayant comme support la radio numérique. Celui-ci permet de diffuser de l'information purement textuelle en utilisant une bande passante très réduite. Journaline utilise son propre langage, le Journaline Markup Language. Celui-ci est en fait un encodage binaire basé sur le XML. Les objets XML, encodés en JML, subissent ensuite une compression spécifique, et sont émis. Une fois reçues par un client du système CONCERT, les données sont décompressées puis reconverties en XML, pour être ensuite traitées.

La figure 3.8 (p.51) illustre l'architecture de CONCERT. Le serveur est chargé de récupérer les données à partir de différentes sources distribuées. Les données peuvent concerner la météo, les attractions à proximité... Les résultats fraîchement obtenus peuvent être de formats différents. Le serveur structure donc le tout et rassemble les informations dans un fichier XML devant respecter la structure définie par le schéma Journaline.

Ce fichier est ensuite broadcasté de manière continue par le *Content server* vers tous les appareils mobiles. Lorsque ceux-ci reçoivent le fichier, celui-ci est décodé et sera utilisé par le *Context manager* de l'application. La figure 3.9 (p.51) détaille l'architecture du client. On

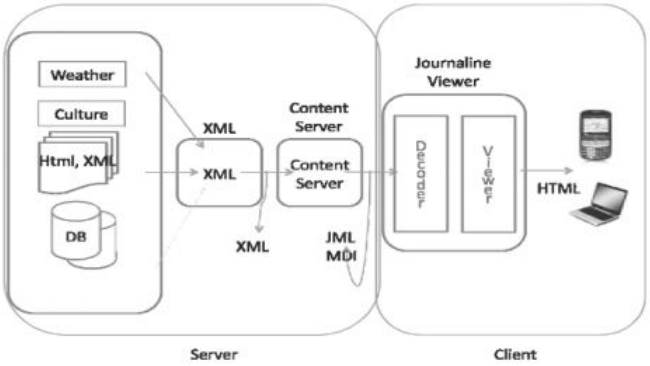

FIGURE 3.8 – *Architecture de CONCERT. Source : [30]*

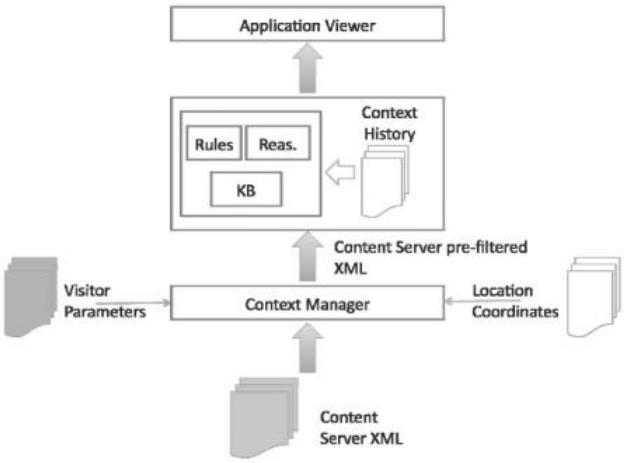

FIGURE 3.9 – *Architecture client de CONCERT. Source : [30]*

remarquera que le *Context manager* est chargé de récupérer toutes les informations nécessaires et que les inférences se font après, du côté client donc.

Outre le choix technologique concernant la communication, placer l'intelligence sur le téléphone de l'utilisateur est une solution problématique. En effet, l'autonomie des smartphones est, encore à l'heure actuelle, un problème majeur et l'utilisation du moteur d'inférence gourmand en ressources risque de compromettre l'expérience utilisateur.

Enfin, les auteurs soulèvent certains problèmes trouvés dans la littérature et tentent de les résoudre par leur recherche.

1. L'absence de consensus sur la notion de contexte en général et dans le cadre mobile en particulier. Pour remédier à ce problème, une nouvelle définition est proposée :

   > *« Context is any relevant information that characterizes the situation of a visitor. A visitor is a traveller taking a trip outside his/her usual environment and her situation is specified by data concerning a) the individual itself, b) the individual's environment (and surroundings) and c) the individual's objective at a particular moment of time. This information can be of use for a computing-application in order to support the visitor's mobility »*

2. L'absence d'un modèle de contexte suffisamment accepté et de méthodes pour la gestion d'informations contextuelles. L'utilisation des réseaux d'ontologies répond à cette problématique. En effet, ceux-ci sont intrinsèquement interopérables. De plus, les ontologies sont propices à l'échange de données et facilitent l'intégration. Sans oublier que les ontologies fournissent une capacité de raisonnement, ce qui est particulièrement intéressant pour l'inférence de données.
3. Le besoin d'une approche scientifique pour étudier le contexte comme une discipline en soi.
4. L'utilisation de senseurs pour rassembler les informations contextuelles pose de sérieuses limitations et nécessite des prérequis pour rendre les systèmes contextuels universellement

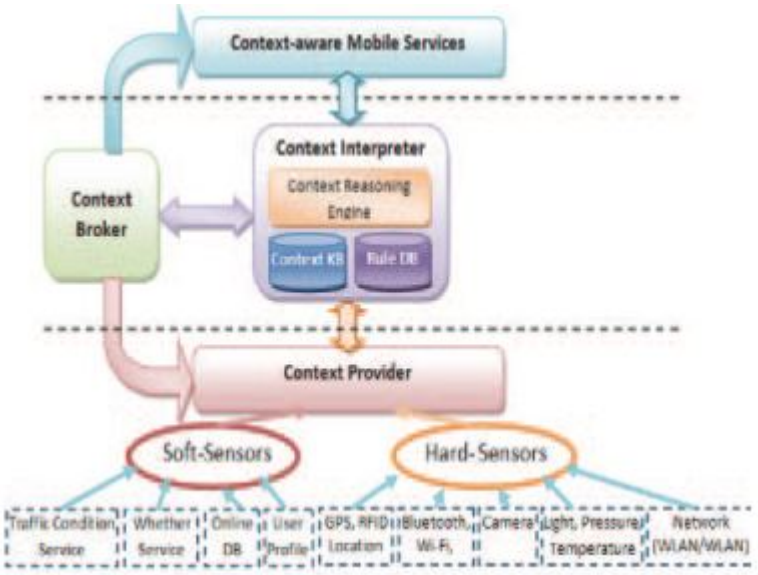

FIGURE *3.10 – Architecture de CAMTON. Source : [39]*

utilisables. C'est pourquoi les auteurs ont décidé de créer une architecture ne nécessitant pas de capteurs externes.

3.10 CAMTON

Saeedi et coll. ont quant à eux fait le choix d'une architecture orientée services (SOA) en créant CAMTON, un *middleware* chargé de fournir des recommandations *context-aware* aux touristes [39]. L'architecture est présentée à la figure 3.10 (p.53) .

Dans un premier temps, le *Context provider* de CAMTON est chargé de collecter les informations de contexte à partir de différents services, sous forme d'ontologies de bas niveau. Ces ontologies sont ensuite transmises au *Context interpreter*, chargé de les relier aux ontologies de haut niveau. Ce *mapping* permet d'uniformiser la structure des informations reçues par les différents services, cette étape est nécessaire pour pouvoir ensuite raisonner.

Le *Context interpreter* contient également une base de données de connaissances et de règles. Elles sont ensuite « mappées » à des ontologies de haut niveau sur lesquelles il est possible de raisonner. L'utilisateur peut, au travers d'une connexion au serveur du *middleware*, formuler une requête pour l'obtention d'un contexte. Le composant *Context Broker* est chargé d'indiquer au *Context Provider* les différentes informations de contexte dont l'utilisateur a besoin.

Un prototype a été créé pour différents téléphones et PDA. L'application assiste le touriste dans la recherche de points d'intérêts situés à proximité et dans la création d'un itinéraire passant par ceux-ci. L'application fournit également de la documentation sous forme audio ou vidéo à propos de l'endroit ou le touriste se trouve.

Néanmoins, ce système nécessite une connexion mobile permanente. Or, nous l'avons déjà précisé, le coût d'une connexion internet mobile est élevé, surtout lorsque l'utilisateur se trouve à l'étranger. De plus, comme beaucoup d'autres applications touristiques, le système est déployé sur l'appareil du touriste. C'est une façon de procéder que nous tenterons d'éviter, afin de soulager au maximum la batterie du téléphone.

3.11 Synthèse

Nous l'avons vu, le développement d'applications touristiques n'est pas un phénomène nouveau. Depuis la fin des années 90, plusieurs recherches ont été menées afin d'améliorer la contextualisation et les systèmes de recommandations dans de telles applications. Nous retrouvons cependant certaines similarités, mais aussi des défauts communs.

La plupart des projets utilisent les ontologies afin de représenter le contexte et celles-ci sont construites de manière à être réutilisables. Nous remarquons également que depuis quelques années, un effort est fourni afin de permettre à différents services de se connecter aux systèmes, rendant ainsi les informations de contexte plus riches et plus précises.

Néanmoins, de nombreux problèmes ont aussi été relevés à plusieurs niveaux et sont synthétisés ci-après.

Problèmes récurrents

Champ d'application réduit La plupart des projets présentés sont intrinsèquement applicables uniquement dans certaines zones, spécialement « préparées ». Pour CONCERT, il est nécessaire de mettre en place une antenne de radio numérique émettant les informations concernant la région. Dans le cas de Cyberguide, c'est l'appareil mobile qui embarque les données concernant les points d'intérêts. Il se retrouve donc limité par la quantité d'information intégrée par les développeurs. Sans compter les balises qui doivent être placées dans le cadre d'une utilisation à l'intérieur d'un bâtiment.

Par ces aspects, ces applications sont ainsi limitées dans leur champ d'application. Bien que réutilisable, leur architecture nécessite un travail important quand elle doit être transposée dans un nouvel environnement.

Intelligence côté client Les recherches effectuées peuvent aussi se séparer en deux catégories, selon l'endroit où est placée l'intelligence du système. Certains préfèrent placer les moteurs d'inférences et/ou systèmes de recommandations sur un serveur, servant d'intermédiaire entre les différents services et les téléphones mobiles des utilisateurs. D'autres en revanche choisissent d'implémenter l'entièreté du système sur l'appareil mobile, évitant ainsi la présence d'un serveur.

Or, nous en avons déjà discuté à plusieurs reprises, cette conception comporte plusieurs points faibles. En effet, l'utilisation de moteurs d'inférence suppose une consommation importante en ressource processeur. Plusieurs effets néfastes peuvent alors apparaître. Premièrement, l'autonomie, point faible principal des appareils mobiles, s'en retrouve réduite. Deuxièmement, sur de larges quantités de données, les calculs d'inférence à la complexité exponentielle peuvent devenir intraitables dans un délai raisonnable pour l'utilisateur. Nous préférons donc un système disposant d'un serveur et libérant le téléphone des processus de calculs, coûteux en énergie.

Cette consommation en ressource peut aussi se manifester, dans une moindre mesure, dans le cadre de la consommation en espace de stockage. En effet, plusieurs solutions proposent de stocker l'intégralité des données d'environnement sur l'appareil mobile. Ce choix, en plus de poser problème pour une application à vocation universelle, pourrait saturer l'espace de stockage disponible sur l'appareil mobile.

Dirigisme trop prononcé Enfin, les évaluations des prototypes ont mené à des observations intéressantes : les utilisateurs n'apprécient pas les systèmes trop intelligents. Ce fut notamment le cas de COMPASS, où les principales remarques négatives concernaient un sentiment de perte de liberté et d'une impression de ne pas avoir accès à toute l'information des environs. Cette problématique a été prise en considération dans la conception de notre système, nous en parlerons dans le chapitre 5 (p.67) .

Méthodes de filtrage Alors que les algorithmes de filtrage collaboratif ont fait leurs preuves depuis longtemps, les auteurs semblent peu enclins à les inclure dans leurs systèmes. Au contraire, les méthodes de filtrage les plus fréquemment utilisées sont celles basées sur l'utilisation d'ontologies.

Cette façon de procéder est justifiée par les auteurs par la nécessité pour l'application de s'adapter au mieux à son contexte d'utilisation. En effet, les ontologies sont particulièrement adaptées pour la modélisation du contexte, comme l'ont montré les auteurs de CONON.

Si l'intention est louable, se limiter uniquement à ce type de filtrage aura parfois certains effets indésirables. Ainsi, le système sera incapable de prendre en compte une information de type *feedback* à propos des différents points d'intérêts. Les points d'intérêts proposés correspondent ainsi de façon très efficace au contexte de l'utilisateur, mais ne peuvent pas être filtrés sur leur qualité intrinsèque, telle que jugée par les utilisateurs.

Ainsi, sur une recherche de restaurant, les systèmes ici présentés se contenteraient de fournir

Projets	Stockage d'informations	Intelligence	Caractère fortement décideur	Filtrage
Cyberguide	Client	Client	Non	Aucun
COMPASS	*Web Services*	Serveur	Oui	*Filtering*, *case-based reasoning*, *item-item filtering*, *category learning*
Flame08	*Web Services*	Serveur	Oui	Ontologies
CONCERT	Radio numérique	Client	?	Réseau d'ontologies
CAMTON	Client/ *Web Services*	Client	?	Ontologies

TABLE *3.1 – Tableau récapitulatif*

une liste parfaitement adaptée au contexte d'utilisation. Ce qui dans notre cas, signifierait une liste de restaurants ouverts au moment de la recherche, à distance raisonnable, adaptés aux goûts de l'utilisateur, etc. Par contre, le système sera incapable de recommander un restaurant plutôt qu'un autre, sur base par exemple des critiques d'autres utilisateurs, ou des choix précédents de l'utilisateur même.

De plus, un tel système ne surprendra jamais un utilisateur, en lui proposant par exemple un point d'intérêt appartenant à une catégorie hors de ses goûts habituels, mais très fréquemment apprécié par des touristes ayant des goûts similaires.

Le lecteur informé aura immédiatement reconnu ici le domaine de prédilection des systèmes de filtrage collaboratifs, présenté dans le chapitre 2 (p.9) . À l'heure où ces systèmes sont utilisés avec succès dans de nombreux domaines connexes, il nous semble approprié de les incorporer dans une solution d'assistance touristique.

Chapitre 4

Modélisation des utilisateurs potentiels

4.1 Introduction

Afin de pouvoir s'orienter dans le choix des fonctionnalités à intégrer à notre système, il est important de cerner les comportements des utilisateurs potentiels. Des observations en situations réelles ont déjà été réalisées par Barry Brown et Matthew Chalmers [17]. Ce chapitre présente les conclusions de leur étude ayant orienté nos choix dans la conception de notre application.

Nous commencerons par présenter les problèmes courants rencontrés par les vacanciers durant leurs séjours. Nous verrons également que ces problèmes n'ont pas forcément un caractère négatif, car leur résolution peut procurer du plaisir aux touristes.

Nous détaillerons ensuite les outils traditionnels utilisés durant leurs voyages et la manière dont ceux-ci sont utilisés pour supporter leur expérience touristique. Par la suite, nous verrons comment les planifications de voyage sont généralement réalisées. En effet, Barry Brown et Matthew Chalmers ont pu observer que celles-ci étaient, la plupart du temps, construites de façon

vague et flexible.

Nous terminerons ce chapitre en détaillant les idées des auteurs sur les possibilités qu'offrent les nouvelles technologies en la matière. Utilisées en collaboration avec les outils plus classiques, celles-ci pourraient améliorer les séjours touristiques.

4.2 Les problèmes des touristes

Les vacanciers sont régulièrement confrontés à un certain nombre de problèmes pouvant être classés dans différentes catégories :

- *Quoi* : Lorsque le touriste arrive pour la première fois sur son lieu de vacances, celui-ci ne connaît généralement pas les points d'intérêts situés dans les environs. Par la suite, il doit faire un choix entre différents types d'activités : visites, shopping, détente... Ces décisions doivent souvent être prises à l'avance et prendre en compte plusieurs paramètres : les temps des trajets et des visites, les conditions météo pour les activités extérieures...
- *Comment* : Sur les sites touristiques, les étrangers doivent être attentifs à leur comportement. Les cultures varient selon les régions, et il est important pour le touriste de pouvoir s'adapter aux us et coutumes locaux.
- *Quand* : Un séjour touristique est limité dans le temps. La planification de la journée est donc une étape importante pour les vacanciers. Ceux-ci doivent coordonner leurs différentes activités selon les heures d'ouverture des attractions, les horaires des transports en commun... De plus, certains points d'intérêts (visites guidées, musées...) nécessitent une réservation à l'avance, ce qui impose une planification rigoureuse de la journée.
- *Où* : Lors de la visite d'une région, les points d'intérêts sont parsemés dans différents endroits, fréquemment inconnus du touriste. Dès lors, il est nécessaire de pouvoir les situer et de réaliser un itinéraire optimal entre ceux-ci.
- *Partage* : Depuis la montée en puissance des réseaux sociaux, de nombreux touristes apprécient le fait de pouvoir partager le plus rapidement possible leurs activités avec leur entourage. Ainsi, la prise et le partage de photos, l'envoi de mails et de cartes postales sont

FIGURE *4.1 – Touristes dans une gare. Source : [17]*

des activités communément appréciées par les vacanciers.

Cependant, ces « problèmes » ne sont pas forcément un aspect négatif des vacances. Chercher un café, se débrouiller dans les stations des transports en commun ou trouver un itinéraire peuvent être en soi des activités appréciables par le visiteur. Fournir une application y répondant de manière inflexible équivaudrait à supprimer une partie du plaisir de la découverte.

En outre, des statistiques américaines montrent qu'en 2010, 76.4% des voyages de loisir impliquent des groupes de deux personnes ou plus [6]. Si cela entraîne des avantages, il est parfois

FIGURE *4.2 – Deux touristes se tournent pour s'orienter. Source : [17]*

FIGURE *4.3 – Touristes pointant vers un endroit et une carte pour les lier ensemble. Source : [17]*

aussi difficile de trouver un consensus satisfaisant les attentes de chacun. Une solution souvent envisagée est alors la séparation temporaire du groupe. Ces choix, séparations et regroupements nécessitent énormément de coordination de la part des membres du groupe. Cela est d'autant plus compliqué que les touristes se déplacent énormément. L'utilisation de technologies telles que les téléphones portables sont dès lors une aide appréciée.

Enfin, il peut être intéressant d'entrer en contact avec d'autres touristes de la même nationalité que le voyageur. Ceux-ci peuvent fournir de bons conseils, dans une langue compréhensible, et sont habituellement une source fiable. Cet aspect est sûrement l'un des plus complexes à réaliser sans une aide permettant aux touristes d'entrer en contact.

4.3 Les outils traditionnels

Barry Brown et Matthew Chalmers ont réalisé plusieurs observations sur la manière dont les outils traditionnels, à savoir les guides touristiques et cartes, étaient utilisés.

Les guides touristiques servent, encore aujourd'hui, de référence pour bon nombre de touristes. Ces guides contiennent une liste des attractions situées dans les environs accompagnées d'une description, et parfois de minicartes. La présentation de chaque activité est standardisée, ce qui permet une compréhension et une comparaison facilitées pour le visiteur. Ainsi, chaque

attraction est traditionnellement accompagnée d'un numéro de téléphone, de ses heures d'ouverture, ainsi que de quelques conseils pratiques. Néanmoins, du fait de l'imprécision de ses cartes, le guide touristique est généralement utilisé en combinaison avec une carte complète de la région visitée.

Concernant les cartes, Barry Brown et Matthew Chalmers ont également observé le fait qu'elles sont rarement utilisées afin de trouver une route optimale entre deux points, mais plutôt pour trouver une direction globale de visite. Parfois, les touristes n'ont pas d'idée précise de l'endroit où ils veulent se rendre ou des activités qu'ils souhaitent réaliser et préfèrent essentiellement flâner dans la ville. De cette façon, ils peuvent s'imprégner de l'atmosphère locale tout en découvrant les lieux au hasard.

Enfin, les auteurs ont relevé le temps passé à observer une carte, pouvant atteindre 20 minutes par consultation. Les cartes sont donc un élément à ne pas négliger, permettant de faire le lien entre les points d'intérêts du monde réel et les guides touristiques.

4.4 Une planification ambigüe

Les auteurs ont analysé la manière dont les touristes planifient leurs séjours. Par la suite, Coccossis et Constantoglou ont également analysé les typologies des touristes, et décrivent un processus de planification similaire [22]. Pour beaucoup, cette étape fait partie intégrante des vacances et est réalisée avec plaisir. Si cette opération se fait essentiellement avant le départ, une partie de la planification peut aussi se faire sur place, avant de commencer la journée. Ces planifications sont cependant rarement précises et contiennent délibérément des imprécisions. Les raisons sont multiples. Tout d'abord, il serait compliqué d'élaborer de manière complète le déroulement du séjour dans un environnement inconnu. Ensuite, un plan imprécis permet de prendre en compte les adaptations de dernière minute.

Signalons aussi que les touristes en partance s'inspirent ou copient des planifications élaborées

précédemment par d'autres personnes. Barry Brown et Matthew Chalmers ont également remarqué que certaines attractions, tels les bus touristiques, pouvaient être utilisées en reconnaissance, de manière à obtenir une vue globale des lieux. Le touriste peut ainsi visiter ultérieurement les endroits qui lui ont semblé particulièrement intéressants de façon plus approfondie. Le bus sert donc ici à fournir de l'information sur la structure des environs et est généralement, aux yeux du touriste, plus efficace qu'un simple plan fortement détaillé.

4.5 Les technologies au service du tourisme

Après avoir étudié les comportements touristiques, Barry Brown et Matthew Chalmers ont présenté quelques idées de technologies pouvant servir aux personnes voyageant à l'étranger. Si les caméras, appareils photo et téléphones portables ont aujourd'hui trouvé leur place aux côtés des outils plus traditionnels, les auteurs proposent des idées d'applications tirant profit des nouvelles technologies et se basant sur les comportements sus-cités.

Sont ainsi mis en avant les mécanismes de partage. En effet, nous avons signalé plus tôt que les touristes apprécient le fait de pouvoir partager leurs séjours avec d'autres personnes qui n'ont pu faire le déplacement. Il serait donc pratique que les applications touristiques soient munies de fonctionnalités permettant l'envoi de photos, de vidéos ou même de cartes postales. Il serait aussi possible d'imaginer des fonctionnalités permettant de mettre en relation des touristes situés au même endroit, ou ayant visité les lieux précédemment, mais ne se connaissant pas. Cela faciliterait entre autres la réutilisation d'itinéraires imaginés par d'autres.

Barry Brown et Matthew Chalmers critiquent également les applications « pop-up », utilisant des notifications pour signaler de manière automatique les points d'intérêts se trouvant à proximité de l'utilisateur. Pour les auteurs, celles-ci possèdent un intérêt limité. En effet, ils statuent qu'une fois arrivé à une attraction, l'environnement fournira des informations plus riches que ce qui peut être offert par un PDA. Si l'utilisation de « pop-up's » est en effet une technique intrusive, donc à éviter, nous ne partageons pas entièrement leur opinion. Nous pensons qu'un

guide peut informer efficacement le visiteur sur son environnement immédiat, dans des domaines desquels il n'est pas expert. Par exemple, en détaillant une oeuvre d'art remarquable dans un musée ou une architecture particulièrement rare dans une église.

Nous terminerons en précisant que la planification est une activité souvent appréciée. Proposer des systèmes effectuant cette étape de manière optimale et automatique n'est donc pas forcément une bonne idée. L'utilisateur pourrait se sentir frustré et ne pas disposer de suffisamment de confiance que pour laisser un programme décider entièrement à sa place.

Les systèmes doivent donc parvenir à se faire une place aux côtés des cartes, guides touristiques et autres outils traditionnels sans vouloir les supplanter. Laisser à l'utilisateur un sentiment de liberté et d'aventure est un aspect important à ne pas négliger.

4.6 Synthèse

Nous l'avons vu, les outils traditionnels en papier sont aujourd'hui encore bien ancrés dans les habitudes des touristes. Changer ces habitudes peut s'avérer difficile, voire impossible. Les applications mobiles doivent en tenir compte et ne pas s'imposer comme un outil de remplacement. Au contraire, elles peuvent être complémentaires, et apporter un plus lors du voyage de l'utilisateur.

Aussi, les problèmes de localisation et de planification ne sont pas des aspects négatifs en soi. Ils font partie intégrante des vacances et leur résolution peut procurer du plaisir.

De plus, les planifications effectuées par les visiteurs restent généralement imprécises, laissant ainsi place à des changements de décisions durant la journée. Une application réglant de manière automatique les problèmes rencontrés risque donc de ne pas être appréciée par les touristes. C'est pourquoi nous préférons développer une application pouvant être vue comme une aide supplémentaire, mais non suffisante. Dès lors, notre système ne procédera pas à la planification d'un itinéraire d'attractions de manière complètement automatique, préférant laisser à certains

moments la main à l'utilisateur.

Chapitre 5

Analyse et Architecture

5.1 Introduction

À partir des problématiques relevées dans le chapitre précédent, nous présenterons ici une architecture d'un système de recommandation touristique contextualisé y répondant. Pour cela, nous commencerons par rappeler la définition du contexte par Dey, et sa division en dimensions. Nous réfléchirons aussi au meilleur moyen de modéliser ces différentes dimensions. Une ontologie existante, SOUPA, a été reprise et adaptée à notre domaine d'application.

Ensuite, nous discuterons de la recommandation contextualisée. Nous rappellerons en quoi consiste le paradigme de *pre-filtering* et comment un système de recommandation peut servir de réponse à des problématiques telles que le *Quoi* et le *Partage*.

Conformément au paradigme du *pre-filtering* et afin de limiter le nombre d'enregistrements à traiter par le moteur de recommandation, des règles de filtrage contextuel ont été construites. Celles-ci permettent de ne sélectionner que les enregistrements de visites effectuées précédemment dans un contexte similaire à celui de l'utilisateur.

Enfin, nous terminerons ce chapitre par une présentation de l'architecture globale de notre système de recommandation contextualisé touristique.

5.2 Contextualisation

5.2.1 Besoin en adaptation

Comme nous l'avons vu à la section 4.2 (p.60) , une application d'assistance touristique conforme aux besoins de ses utilisateurs doit répondre à une série de problématiques :

- *Quoi* : Le système doit être capable de proposer au touriste des points d'intérêts correspondant aux mieux à ses préférences. Pour ce faire, il est nécessaire de posséder suffisamment d'information sur ces points d'intérêts et de structurer cette information de façon à permettre le raisonnement sur celle-ci. Il est également nécessaire de posséder une quantité d'informations suffisante à propos de l'utilisateur, correspondant ainsi à une nouvelle dimension, le *Qui*.
- Quand : Le système doit être capable de prendre en compte la dimension temporelle. Il doit pouvoir gérer cette dimension de façon à supporter la planification du séjour touristique. Il doit pouvoir considérer aussi bien l'aspect pratique (horaires d'ouvertures, durée des visites...) que les habitudes de l'utilisateur (préférence pour la plage le matin).
- Où : le système doit être capable de prendre en compte la dimension physique. Cela comprend principalement le support de la localisation géographique, mais peut également intégrer d'autres informations sur l'environnement telles que les conditions météorologiques.
- Partage : le système doit être capable de mettre en relation ses différents utilisateurs. Le partage d'information devant permettre d'améliorer l'efficacité globale du système.

Remarquons que ces problématiques coïncident fortement avec les quatre catégories d'information associées à la définition standard du contexte donnée par Dey (cf. 2.2.2 (p.11)). Un système capable de s'adapter à ces différentes dimensions est donc, par définition, un système contextualisé. Chaque problématique peut ainsi être considérée comme une dimension du contexte d'utilisation de l'application.

C'est pourquoi la majorité des systèmes touristiques déjà réalisés ces dernières années, tels que présentés dans le chapitre 3 (p.33) , utilisent déjà des mécanismes de contextualisation.

Dans les sections suivantes, nous détaillerons les techniques choisies pour supporter cette contextualisation.

5.2.2 Les ontologies comme support d'adaptation

Introduction

Comme nous l'avons vu précédemment à la section 2.5 (p.25) , une étude réalisée par Strang et Linnhoff-Popien [42] a montré que les ontologies sont l'outil le plus adéquat pour la gestion des informations de contexte. Depuis, comme étudié dans le chapitre 3 (p.33) , de nombreuses recherches se sont inspirées de cette étude pour proposer des systèmes contextualisés utilisant des ontologies comme support de la gestion du contexte.

Nous retiendrons particulièrement l'étude 3.6 (p.43) de Harry Chen et coll. [21] dans laquelle ils proposent une ontologie standard pour la représentation des informations de contexte et le raisonnement sur celles-ci. C'est cette ontologie qui a été choisie comme base pour notre système. En effet, plusieurs recherches comparatives sur les ontologies destinées aux systèmes contextualisés ont été menées et ont relevés les bons résultats de SOUPA (Ye et coll. [46] ;Baumgartner et Retschitzegger[15]).

L'ontologie de base : SOUPA

Dans la première étude, Ye et coll. [46] comparent plusieurs ontologies de contexte sur différents critères, parmi lesquels le respect des bonnes pratiques de conception des ontologies. Chaque ontologie a été jugée sur ce critère, comme montré à la figure 5.1 (p.70) .

De plus, le tableau 5.2 (p.70) indique les thèmes ciblés par les différentes ontologies. Nous remarquons que SOUPA possède toutes les dimensions nécessaires pour notre système (cf. 5.2.1

Ontology-based Models	CoBrA (SOUPA)	Gaia	GLOSS	ASC (CoOL)	CONON
Coherence	√				
Clarity	√			√	√
Extensibility	√	√	√	√	√
Ontological Commitment	√		√		
Orthogonality	√		√		
Encoding Bias	√	√	√		√

FIGURE *5.1 – Évaluation des ontologies sur base du respect des bonnes pratiques de conception. Source : [46]*

Ontology-based Models	CoBrA (SOUPA)	Gaia	GLOSS	ASC (CoOL)	CONON
Location	√		√	√	√
Person/Agent	√		√		√
Time	√		√		
Activity	√		√	√	√

FIGURE *5.2 – Évaluation des ontologies sur base des thèmes abordés. Source : [46]*

(p.68)), à savoir *Location* (Quoi,Où), *Person/Agent* (Qui, Partage), *Time* (Quand) et *Activity* (Quoi, Quand).

L'ontologie étendue

L'ontologie étendue a été conçue pour être adaptée à un système touristique. Pour cela, elle reprend certaines classes de SOUPA présenté précédemment.

Tout d'abord, la classe *Person* sera reprise afin de décrire les utilisateurs et de contenir leurs informations personnelles. Du raisonnement pourra alors être effectué lors de la contextualisation, en catégorisant les utilisateurs selon leur âge par exemple.

Les classes *Location*, *Space* (chargées de fournir les concepts de localisation et le vocabulaire associé) et *Time* seront également reprises de SOUPA.

Ensuite, nous y avons ajouté certains concepts manquants, mais nécessaires pour une représentation complète du contexte dans le thème touristique. Ainsi, nous avons créé une ontologie *Weather*, chargée de fournir les aspects météorologiques. Elle est illustrée à la figure 5.3 (p.71)

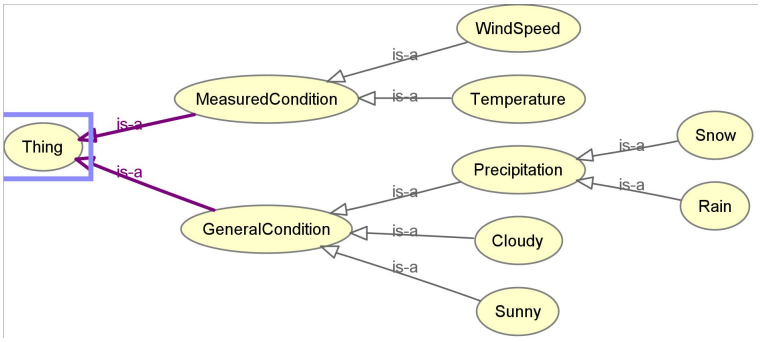

FIGURE 5.3 – *L'ontologie Weather*

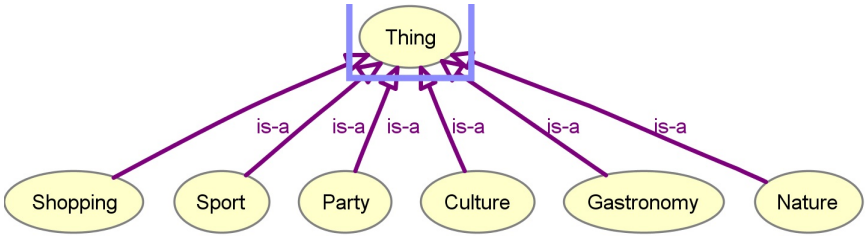

FIGURE 5.4 – *L'ontologie Interest*

et contient des attributs tels que la température et la vitesse du vent. *GeneralCondition* permet de représenter les conditions météo de manière plus générale, avec une séparation des conditions selon la présence ou non de précipitations. Cette ontologie reste simple car le domaine du tourisme ne requiert pas plus de précision concernant les conditions météorologiques (contrairement au domaine de l'aviation, par ex.).

L'ontologie *Interest* a été créée afin de décrire les différentes catégories d'intérêts. On retrouve donc *Shopping*, *Sport*, *Party*, *Culture*, *Gastronomy* et *Nature*, comme illustré à la figure 5.4 (p.71) .

Enfin, la classe *Visit* reprend le concept de meeting présent dans SOUPA en l'adaptant. Son rôle est de représenter une visite en l'associant aux autres entités précitées. Une visite possédera donc des propriétés telles que *hasWeather*, *hasPeriod*, *hasInterest*...

Les ontologies présentées ici peuvent être étendues, leur rôle n'étant pas d'être exhaustives. Elles peuvent donc être considérées comme un *framework* pour la création d'ontologies pour

Source	Classes
SOUPA	Person
	Location
	Space
	Time
Extensions	Weather
	Interest
	Visit

TABLE *5.1 – Ontologie étendue*

notre système. Le tableau 5.1 (p.72) synthétise les différentes ontologies utilisées.

5.3 La recommandation contextualisée

5.3.1 Introduction

Dans la section 5.2 (p.68) , nous avons présenté les procédés mis en oeuvre afin d'obtenir des points d'intérêts correspondant au contexte actuel de l'utilisateur. Cependant, parmi les points d'intérêts correspondants au contexte, il est actuellement impossible de connaître ceux correspondant le mieux aux préférences de l'utilisateur. Jusqu'à présent, la problématique du *Quoi* n'a donc pas été entièrement résolue.

Comme présenté à la section 2.3 (p.12) , trier ou sélectionner une liste d'objets selon les préférences d'un utilisateur est la raison d'être des systèmes de recommandation.

Pour rappel, les systèmes de recommandation contextualisés, ou CARS (*Context-Aware Recommendation System*), combinent les techniques de recommandation et de contextualisation. Plusieurs techniques existent et ont été présentées à la section 2.4 (p.20) . Nous avons aussi montré dans cette section que les CARS étaient plus efficaces que les systèmes de recommandation traditionnels (bidimensionnels).

C'est pourquoi un CARS est au centre de notre système pour fournir des recommandations de

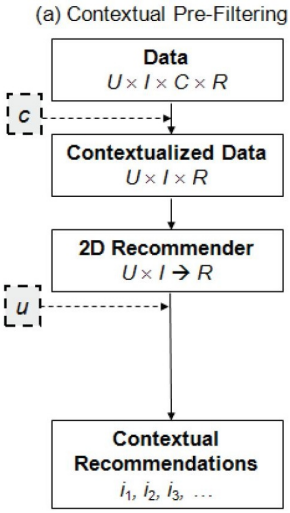

FIGURE 5.5 – *Contextual pre-filtering. Source : [8]*

points d'intérêts à nos utilisateurs. Dans les sections suivantes, nous présenterons les différentes parties de l'architecture de ce CARS.

5.3.2 Le paradigme du pre-filtering

Parmi les trois techniques existantes, *contextual pre-filtering*, *contextual post-filtering* et *contextual modeling*, nous avons décidé de nous concentrer sur la première. En effet, comme discuté dans le chapitre 2 (p.9) , les deux premières permettent l'utilisation de systèmes de recommandation traditionnels.

Dans le cas du *contextual pre-filtering*, illustrée à la figure 5.5 (p.73) , l'utilisation du contexte se fait au début du processus, limitant ainsi l'ensemble des objets qui seront traités. Seuls les objets correspondant au contexte actuel sont donc gardés, et une note leur sera attribuée par un système de recommandation traditionnel. Contrairement à la technique de *post-filtering*, le fait de limiter dès le départ l'ensemble des objets qui seront traités permet d'alléger la charge de travail du système, le rendant plus efficace.

Après traitement, nous obtiendrons donc des points d'intérêts correspondant au contexte actuel de l'utilisateur, mais triés selon ses préférences. Dès lors, proposer cette liste triée permet de satisfaire au mieux les demandes des utilisateurs, répondant ainsi à la problématique visée, le *Quoi*.

En utilisant le paradigme du *pre-filtering* pour notre CARS, il devient nécessaire de concevoir un processus de filtrage contextuel sur les données d'entrée de notre système. Concrètement, il faut d'abord établir les dimensions contextuelles sur lesquelles le filtrage sera réalisé. Nous avons retenu les dimensions suivantes, correspondant aux différentes dimensions de notre ontologie : *Person*, *Location*, *Time*, *Weather* et *Interest*.

Comme l'ont montré Adomavicius et Tuzhilin [8], l'efficacité d'un CARS est déterminée en grande partie par la sélection de ses règles de filtrage contextuel. Présentons quelques règles permettant de représenter le contexte d'un objet. L'objectif ici n'est pas de proposer une liste exhaustive des règles définies, mais d'en proposer un échantillon représentant les idées principales. Tout comme les dimensions contextuelles, les règles de filtrage présentées ne représentent donc qu'un *framework* pour la conception du système, la définition d'un ensemble précis de règles relevant plutôt du paramétrage.

Il est par exemple possible de filtrer selon le profil des utilisateurs. Connaissant l'âge de la personne, nous pouvons garder les points d'intérêts visités par des touristes de la même tranche d'âge uniquement. Dans le cas d'un utilisateur ayant une trentaine d'années, nous obtiendrions la règle suivante :

$$(?v\ visitedBy\ ?u)\ \wedge\ (?u\ hasAge\ ?a)\ \wedge\ (?a\ lessThan\ \text{"40"})\ \wedge\ (?a\ moreThan\ \text{"20"})$$
$$\Rightarrow\ (?v\ selected\ \text{"}true\text{"})$$

Ainsi, après l'application de cette règle, l'ensemble des enregistrements retenus pour la recommandation ultérieure ne contiendra que les visites ayant été réalisées par des touristes entre

20 et 40 ans.

Concernant la localisation, il serait possible de ne conserver que les enregistrements concernant les points d'intérêts se situant à proximité. Selon le rayon souhaité par l'utilisateur et sa position actuelle (« loc »), il est dès lors possible de calculer les distances des différents points d'intérêts, comme le montre la règle suivante :

$$(?v\ hasLocation\ ?l) \wedge (?d\ fromLocation\ "loc") \wedge (?d\ toLocation\ ?l) \wedge (?d\ lessThan\ "2km") \Rightarrow (?v\ selected\ "true")$$

La syntaxe utilisée ici ne respecte pas entièrement les classes et propriétés proposées par SOUPA. En effet, l'objectif ici est de présenter les règles et les buts qu'elles souhaitent atteindre. Présenter des règles réelles utilisables dans SOUPA rendrait leur compréhension ardue, vu la verbosité de la syntaxe du OWL DL.

La dimension du temps permet d'identifier les périodes pendant lesquelles les points d'intérêts sont visités. À titre d'exemple, il est improbable qu'un utilisateur désire se rendre dans une boîte de nuit durant la journée. Il est dès lors possible d'imaginer une règle qui, à partir de l'heure actuelle, ignore les enregistrements possédant un contexte avec une heure différente. Nous obtenons alors, lors d'une recherche effectuée en journée :

$$(?v\ hasTime\ ?t) \wedge (?t\ isDay\ "true") \Rightarrow (?v\ selected\ "true")$$

Dans ce cas, seules les visites ayant été effectuées en journée seront conservées. Dans le même ordre d'idées, il est possible de créer des règles similaires prenant en compte la saison, le mois...

Enfin, il serait intéressant de ne pouvoir conserver que les enregistrements dont le contexte météo correspond aux conditions actuelles. Cela permettrait d'éliminer les visites effectuées par temps de pluie alors qu'il fait ensoleillé, et réciproquement. La règle suivante illustre ce procédé :

$$(?v\ hasWeather\ ?w) \wedge (?w\ hasCondition\ "sunny") \Rightarrow (?vselected\ "true")$$

Il serait également possible d'ajouter des règles prenant en compte la température à la place de la condition. On ignorerait alors les enregistrements ayant un contexte météo ne possédant pas une température similaire à celle relevée lors de la requête.

Nous avons parcouru ici quelques règles illustratives de la première étape du paradigme de *contextual pre-filtering*. Après application de ces règles, il ne subsiste que les enregistrements correspondant au contexte actuel, filtrés grâce aux règles. Ces enregistrements seront ensuite utilisés comme base pour le processus de recommandation que nous allons détailler ci-dessous.

5.3.3 Le système de recommandation

La dernière problématique n'ayant pas été abordée est la question du « Partage ». Celle-ci pourra être traitée par le système de recommandation, en plus des problématiques du « Quoi » et du « Qui ».

Rappelons que la technique de *pre-filtering* permet l'utilisation de systèmes traditionnels (bi-dimensionnels), présentés à la section 2.3 (p.12) . Dans cette partie, nous avions discuté des différentes techniques existantes. Celle retenue ici sera l'algorithme *user-based*, membre de la famille des des systèmes de recommandation collaboratifs.

En effet, l'avantage principal des systèmes collaboratifs était de prendre en compte l'historique de tous les utilisateurs pour effectuer une recommandation. Cela permet de mettre en relation deux individus ne se connaissant pas, mais possédant des goûts similaires.

De plus, contrairement à l'algorithme *item-based*, son homologue *user-based* ne repose pas sur la similarité des objets. Cette spécificité permet de recommander une variété plus large d'objets, et d'ainsi augmenter les chances de découvertes pertinentes pour l'utilisateur.

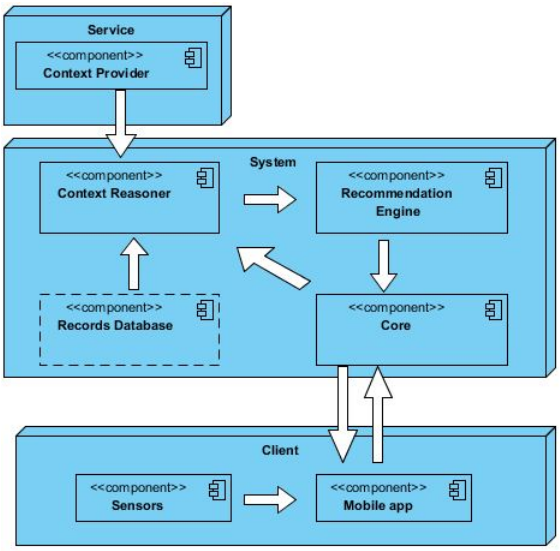

FIGURE *5.6 – Architecture du système touristique*

Nous observons ici que la mise en place de ces techniques peut répondre au concept de « Partage » : l'historique des utilisateurs est en quelque sorte partagé (puisqu'utilisé dans son entièreté à chaque recommandation) afin de pouvoir recommander des objets à un utilisateur. Les objets recommandés peuvent alors être fort différents de ceux présents dans l'historique de l'utilisateur, mais ont été appréciés par de nombreux utilisateurs similaires. Ainsi, chaque notation d'un utilisateur particulier permet d'améliorer la précision de l'ensemble du processus de recommandation, pour l'ensemble des utilisateurs.

5.4 L'architecture

Dans les sections précédentes, nous avons discuté les différents aspects intégrés dans le système touristique. Nous présentons ici l'architecture globale et les interactions entre les différents composants. L'architecture est divisée en trois parties et est illustrée à la figure 5.6 (p.77) .

5.4.1 Client

La partie *Client* est constituée d'une application mobile *Mobile app*, recevant l'information des senseurs *Sensors* du téléphone (localisation GPS...). L'application est chargée d'afficher les résultats, d'envoyer les requêtes et d'envoyer les *feedbacks*, mais ne contient aucune intelligence. Ceci pour les raisons évoquées à la fin du chapitre 3 (p.33) : être disponible pour les téléphones les moins puissants et économiser au maximum leur autonomie.

5.4.2 Système

Le système est composé d'un noyau, appelé *Core*, chargé de communiquer avec les clients. Il reçoit les requêtes et les envoie au *Context Reasoner*, chargé de sélectionner les enregistrements (visites effectuées par l'ensemble des utilisateurs du système possédant un contexte similaire) stockés dans *Records Database*.Cette base de données peut être implémentée soit localement, soit être fournie par un tiers extérieur au système.

Le sous-ensemble des enregistrements sélectionné est ensuite envoyé au *Recommendation Engine*. Celui-ci attribuera une note pour chaque enregistrement, sur base d'un algorithme de *collaborative filtering*. Il créera ainsi une liste triée selon les préférences de l'utilisateur à l'origine de la requête. Cette liste sera ensuite fournie au *Core* qui se chargera de la transmettre à l'appareil mobile.

Le composant *Core* est également capable de recevoir les *feedbacks* donnés par les utilisateurs ayant terminé la visite d'un point d'intérêt. Il les enregistrera dans la base de données *Records Database* pour qu'ils puissent être utilisés lors de recommandations ultérieures.

5.4.3 Service

La partie *Service* est constituée du *Context Provider*. Ces services sont les systèmes tiers chargés de fournir le *Context Reasoner* en informations de contexte.

5.5 Réponse aux problèmes récurrents

Dans le chapitre 3 (p.33) , nous avons relevé dans la section 3.11 (p.55) les problèmes récurrents dans les solutions existantes. Nous présentons ici nos solutions et moyens mis en oeuvre afin de les éviter.

Champ d'application réduit L'architecture a été adaptée de manière à pouvoir fonctionner avec des bases de données de points d'intérêts externes. Cette façon de faire permet au système d'être utilisé n'importe où, pour peu que des points d'intérêts soient disponibles et accessibles de manière structurée. Ainsi, le système devient flexible et adaptable.

Intelligence côté client Notre architecture a été pensée de manière à soulager au maximum l'appareil mobile, limité à effectuer principalement du simple affichage. Le restant étant effectué par un serveur, communiquant avec les différents services et les appareils des clients. En suivant notre approche, les opérations réseau et de localisation du mobile peuvent aussi être réduites de façon drastique, puisque le client n'est pas conçu pour être utilisé en permanence, sauf lors du guidage. L'autonomie des appareils devrait donc être préservée et la plupart d'entre eux devraient être capables de supporter notre client.

Dirigisme trop prononcé Réaliser un système de support touristique trop dirigiste est un écueil courant. Afin d'éviter cela, il est nécessaire de laisser à l'utilisateur un sentiment de liberté et d'accès à toute l'information. C'est pourquoi les réponses aux requêtes ne se limitent pas au seul point d'intérêt calculé comme étant le plus pertinent, mais sont constituées d'une liste triée. Ainsi, l'utilisateur a un accès direct à un ensemble d'objets correspondants au contexte actuel et peut, s'il le souhaite, les prendre en compte lors de la création de son itinéraire.

Méthodes de filtrage Comme présenté dans ce chapitre, notre système de recommandation contextualisé utilise la technique dite de *pre-filtering*, permettant l'utilisation conjointe d'un système de recommandation classique (bidimensionnel) et d'un filtrage contextuel. Il est dès lors possible de combiner les avantages issus de la contextualisation grâce aux ontologies et ceux issus

des systèmes de recommandation traditionnels, et en particulier des techniques de *collaborative filtering*.

5.6 Conclusion

Nous avons présenté dans ce chapitre notre architecture de système de recommandation touristique contextualisé. Ainsi, nous avons décidé d'utiliser les technologies de contextualisation et de recommandation pour répondre aux problématiques rencontrées par les touristes (*Quoi, Qui, Quand, Où, Partage*), car celles-ci correspondaient aux dimensions du contexte définies par Dey.

Ensuite, nous avons abordé la manière de combiner les technologies de recommandation et de contextualisation dans un *Context-Aware Recommender System* (CARS). Nous avons expliqué notre choix du paradigme du *pre-filtering* basé sur la possibilité d'utiliser les systèmes de recommandation traditionnels et sa moindre complexité. Nous avons aussi justifié l'utilisation d'un algorithme de filtrage collaboratif, permettant de répondre aux problématiques du *Quoi, Qui* et du *Partage*.

Au final, nous avons présenté une architecture complète permettant d'assembler ces différentes techniques pour produire un système fonctionnel d'assistance touristique contextualisé.

Dans le chapitre suivant, nous introduirons SmartTrip, un prototype basé sur cette architecture et démontrant sa faisabilité réelle.

Chapitre 6

Application

6.1 Introduction

Dans le chapitre précédent, nous avons introduit une architecture d'un système de recommandation touristique contextualisé. Un prototype a été réalisé, démontrant ainsi sa faisabilité. Ce chapitre a pour objectif de le présenter.

Pour ce faire, nous commencerons par décrire les objectifs visés par notre prototype, nommé SmartTrip. Nous aborderons ensuite les exigences fonctionnelles et non-fonctionnelles de l'application. Pour chacune, nous expliquerons nos motivations et en quoi elles peuvent s'avérer utiles pour le touriste et la résolution de ses problématiques. Nous préciserons également les informations nécessaires pour le bon fonctionnement du système.

Nous détaillerons ensuite les cas d'utilisation de SmartTrip. Les différentes étapes de ces cas seront également présentées et illustrées par des captures d'écran du prototype développé.

Ensuite, nous aborderons l'architecture détaillée de SmartTrip. Nous passerons en revue les différents composants présents et leurs interactions seront mises en évidence par le biais d'un

diagramme de séquence.

Enfin, nous terminerons par les choix technologiques qui ont pu être faits lors du développement du système.

6.2 Objectifs

Nous nous intéressons ici aux buts que le système tend à atteindre. Le goal diagram de la figure 6.1 (p.83) nous permet d'exprimer de manière informelle la hiérarchie des buts visés par le système et les moyens qui vont être mis en place afin d'atteindre ces objectifs.

L'objectif de ce prototype est de proposer une solution correspondant à l'architecture du chapitre 5 (p.67) . Pour cela, SmartTrip répondra aux objectifs présentés sur le diagramme 6.1 (p.83) . Supporter la recherche d'activités adaptées ainsi que la planification des visites et guider les touristes sont donc les fonctionnalités principales de SmartTrip.

Concernant les activités adaptées, celles-ci devront répondre aux préférences de l'utilisateur, ce qui sera mis en oeuvre par l'utilisation d'un algorithme de recommandation et l'implémentation d'une recherche de points d'intérêts par catégorie. De même, l'application devra être en mesure de prendre en compte le reste du contexte. Pour cela, le prototype pourra ne recommander que des points d'intérêts correspondant aux conditions météorologiques actuelles.

De plus, l'application disposera d'un module de création d'itinéraires. L'utilisateur pourra créer une liste de points d'intérêts de son choix et les ordonner selon l'ordre dans lequel il souhaite les visiter.

Enfin, l'application sera capable de guider le touriste jusqu'à ses activités. Concrètement, celles-ci seront affichées sur une carte des environs, avec l'itinéraire vers les attractions.

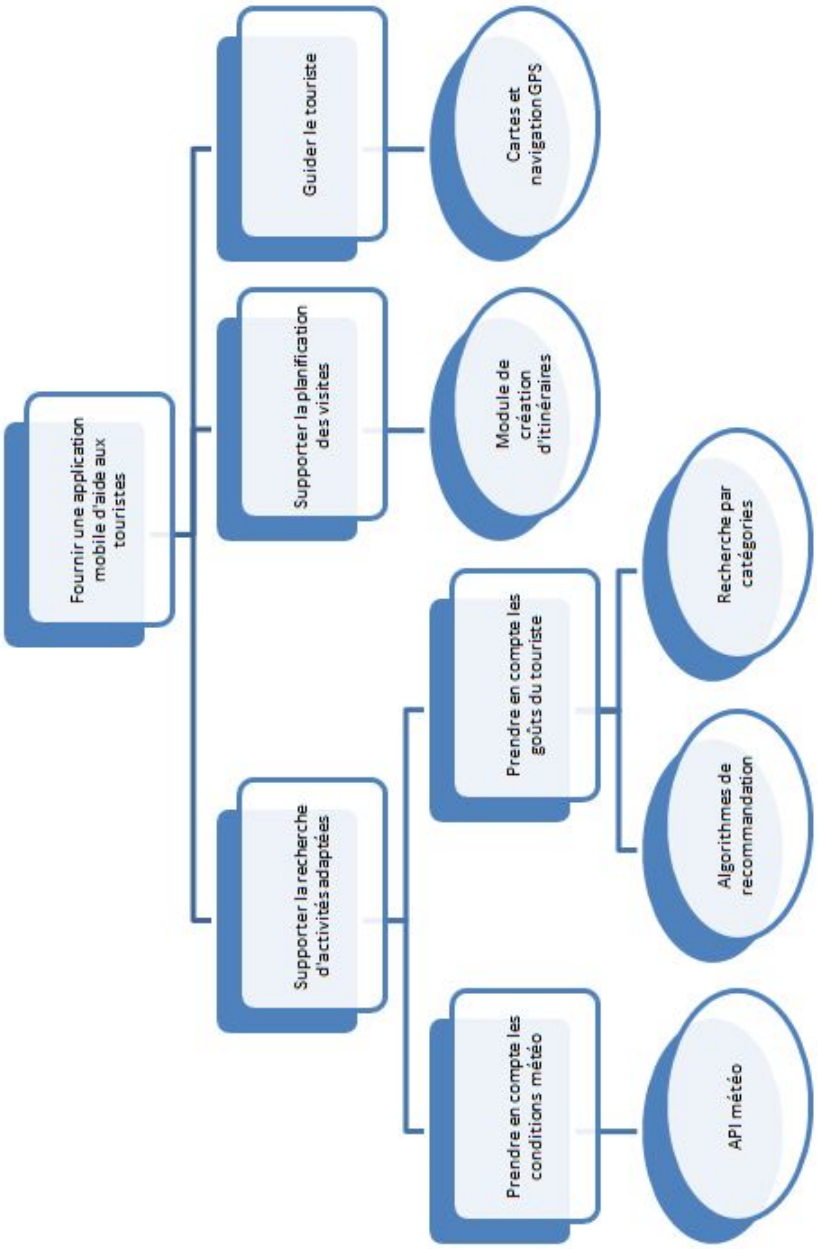

FIGURE 6.1 – *Diagramme de buts*

6.3 Exigences

6.3.1 Exigences fonctionnelles

Notre système, SmartTrip, se destine à être une assistance touristique contextualisée. Sur base de l'architecture présentée dans le chapitre 5 (p.67) , nous avons constitué une sélection limitée de fonctionnalités, permettant de répondre partiellement aux problématiques présentées dans les chapitres précédents au sein d'un prototype à développement rapide. Ces fonctionnalités sont présentées ci-dessous :

Une recherche par catégories

Tous les utilisateurs n'ont pas les mêmes préférences. Si certains préfèrent les activités culturelles, d'autres souhaiteraient s'orienter vers des attractions plus sportives, ou gastronomiques. Pour chaque requête, SmartTrip proposera donc à l'utilisateur de choisir les différentes catégories d'activités pouvant l'intéresser. Nous avons relevé les catégories suivantes :

- Gastronomie
- Culture
- Nature (parcs, points de vue...)
- Fêtes (dancing, soirées...)
- Shopping
- Sport

Cette approche se différencie d'autres projets proposant soit une approche plus globale (proposer tous les points d'intérêts quelle que soit leur catégorie), soit plus dirigiste (ex. : ne proposer que des restaurants aux alentours des heures de table).

De plus, une recherche par catégorie permet d'effectuer un premier filtrage, aidant ainsi à la résolution d'une des problématiques rencontrées par le touriste, le *Quoi*.

Prise en compte de la localisation

L'application sera capable de détecter précisément la position de l'utilisateur, et proposera une recherche limitée à un certain rayon autour de cette position, précisé par l'utilisateur. Aucun point d'intérêt se situant à une distance supérieure à celle indiquée par l'utilisateur ne sera donc proposé par la suite. Cet aspect permet de contribuer à la question du *Où*.

Prise en compte des conditions météo

Afin de prendre en compte le contexte dans lequel se trouve l'utilisateur, SmartTrip sera capable de prendre en compte les conditions météo pour générer sa liste de points d'intérêts. Ainsi, elle ne proposera pas d'activités extérieures en cas de conditions peu clémentes.

Afin de conserver le caractère flexible de notre application, cette fonctionnalité prendra la forme d'une option, pouvant être désactivée à tout moment par l'utilisateur.

Présentation triée de points d'intérêts

Après la sélection des catégories, SmartTrip recherchera, selon l'historique de l'utilisateur, les points d'intérêts les plus adaptés aux goûts du touriste. Le résultat sera présenté sous la forme d'une liste triée, en commençant par les activités correspondant au mieux au profil de l'utilisateur.

De plus, l'utilisation d'une liste proposant un ensemble d'activités, permettra à l'utilisateur de conserver un sentiment de liberté et d'avoir accès à toute (ou du moins à une grande partie de) l'information.

Création d'un planning

Après la lecture de la liste des résultats sur son appareil mobile, l'utilisateur aura la possibilité de créer le planning de la journée. Pour ce faire, il lui suffira de choisir l'ordre des attractions qu'il souhaite visiter en réordonnant la liste des résultats via un système de « drag and drop ».

Nous en avions discuté dans le chapitre 4 (p.59) , la planification est une étape réalisée par beaucoup de touristes. Cette fonctionnalité vise donc à fournir une aide permettant une planification, imprécise certes, du séjour touristique.

Guider l'utilisateur à travers les points d'intérêts

Sans surprise, la carte est, avec le guide touristique, l'un des instruments les plus utilisés par les vacanciers. Afin d'aider les utilisateurs à s'orienter et à se diriger parmi les différents points d'intérêts préalablement sélectionnés, SmartTrip sera capable de les positionner sur une carte. Il pourra également générer l'itinéraire idéal entre ceux-ci.

Donner un *feedback* au système

Lorsque l'utilisateur a créé son itinéraire, celui-ci peut le communiquer au serveur. La liste des points d'intérêts est alors enregistrée dans son historique et sera prise en compte pour améliorer la précision de la recherche des utilisateurs similaires du moteur de recommandation.

6.3.2 Exigences non fonctionnelles

Universalité

SmartTrip devra être capable de fonctionner dans le monde entier, sans restriction de région. Contrairement à d'autres prototypes existants, nous souhaitons éviter l'utilisation de matériel spécifique dans l'environnement. Nous souhaitons également nous concentrer sur des technologies déjà démocratisées auprès du grand public.

Qualité

SmartTrip devra fournir des listes d'attractions pertinentes aux yeux de ses utilisateurs. L'analyse des profils et la détection des préférences doit donc se faire de manière minutieuse en utilisant les algorithmes de recommandations adéquats.

Compatibilité

Nous souhaitons que notre application soit utilisable pour le plus grand nombre de personnes. L'application mobile en elle-même se limitera à l'envoi des requêtes et à l'affichage des résultats. Cela permet une exploitation minimum du processeur du téléphone, et garantit que même les smartphones les moins puissants posséderont les ressources nécessaires pour faire fonctionner l'application.

6.4 Cas d'utilisation

Nous présentons ici les différents scénarios implémentés dans SmartTrip. Ceux-ci sont présentés à la figure 6.2 (p.88) . Par le biais de capture d'écrans de l'application, nous illustrerons les différentes étapes devant être réalisées par l'utilisateur.

Premièrement, et afin de pouvoir accéder à la base de données des points d'intérêts de Foursquare, l'utilisateur doit posséder un compte de ce réseau social et se connecter. Une capture d'écran de cette étape sur SmartTrip est présentée à la figure 6.8 (p.91) . Il peut ensuite demander la création d'une liste de points d'intérêts. Ce scénario est présenté à la figure 6.3 (p.89) .

Les captures d'écrans 6.4, 6.5, 6.6, 6.7 (p.90) et 6.9 (p.91) décrivent les différentes étapes de SmartTrip. Dans la figure 6.4 (p.90) , l'utilisateur sélectionne les catégories qui lui semblent pertinentes. Il précise également le rayon dans lequel les points d'intérêts doivent se trouver et s'il souhaite que la prise en compte des conditions météo actuelles soit activée.

Lorsque la requête de l'utilisateur a été traitée par le serveur, l'application SmartTrip affiche la liste des points d'intérêts retenus (cf. figure 6.5 (p.90)). L'utilisateur peut alors sélectionner ceux qui l'intéressent.

Ensuite, l'utilisateur peut ordonner sa liste de points d'intérêts afin de créer son itinéraire personnalisé. Cette étape est présentée aux figures 6.6 et 6.7 (p.90) . Lorsque l'itinéraire a été créé, l'utilisateur peut l'envoyer au serveur. Celui-ci l'enregistrera et s'en servira lors du calcul

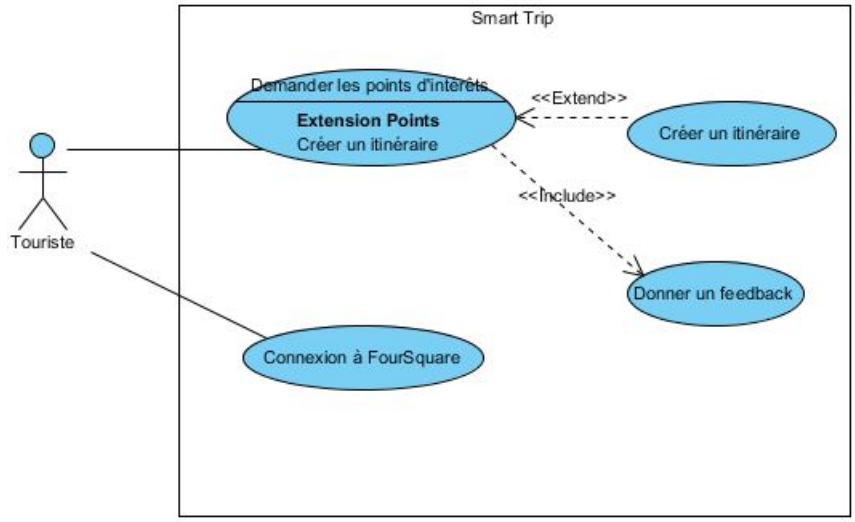

FIGURE *6.2 – Diagramme des cas d'utilisation*

des recommandations futures.

Enfin, l'utilisateur peut afficher son itinéraire sur une carte obtenue grâce au service Google Maps 6.9 (p.91) .

6.5 Architecture du prototype

6.5.1 Diagramme des composants

La partie client

Le composant *Display* permet la gestion de l'affichage des interfaces sur le téléphone de l'utilisateur. Celui-ci communique avec le *RequestManager*, chargé de gérer les requêtes du client. Il est donc capable d'obtenir des informations du *LocationManager* destiné à récolter les informations des différents capteurs de localisation (puce GPS, boussole intégrée...) et du composant *MapManager*.

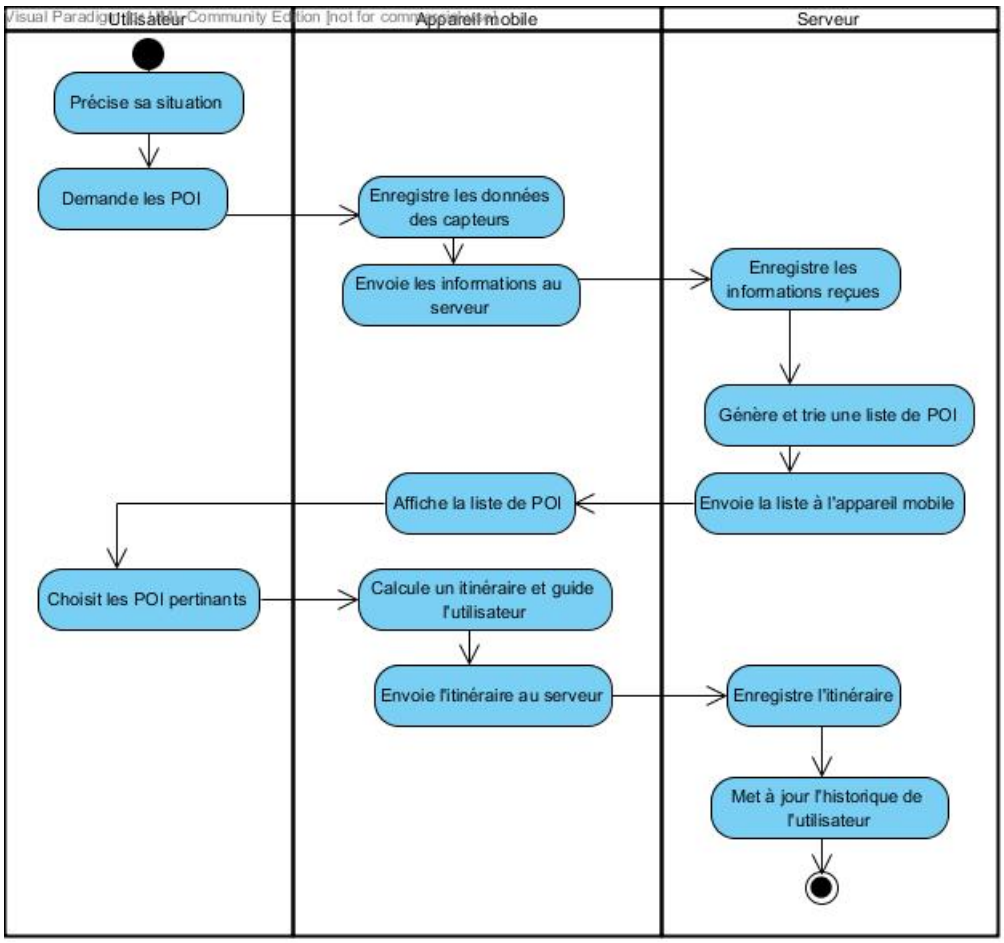

FIGURE 6.3 – *Scénario : créer une liste de points d'intérêts*

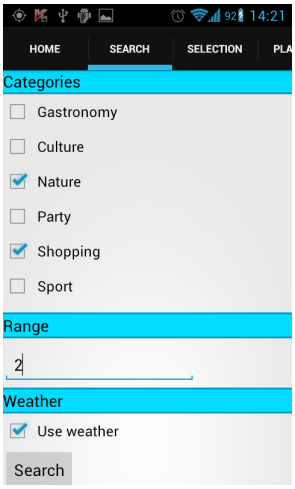

FIGURE 6.4 – *Formulaire de recherche*

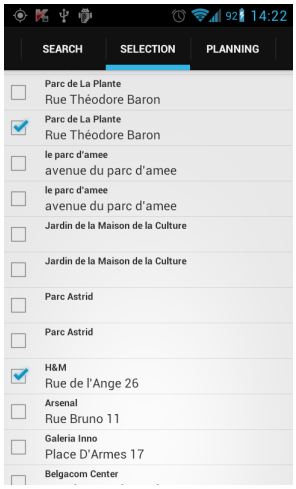

FIGURE 6.5 – *Affichage des résultats*

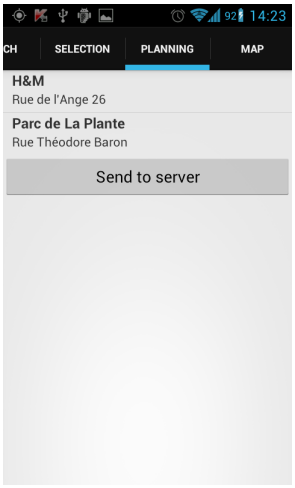

FIGURE 6.6 – *Affichage des PoI retenus par l'utilisateur*

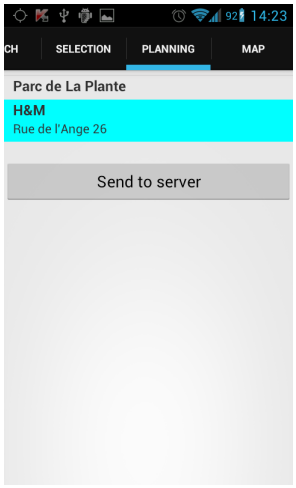

FIGURE 6.7 – *Modification de l'ordre des PoI par un système de « Drag and Drop »*

FIGURE 6.8 – *Connexion au service Foursquare*

FIGURE 6.9 – *Affichage de l'itinéraire sur Google Maps*

Le composant *MapManager* est chargé de générer la carte, sur laquelle l'itinéraire contenant les différents points d'intérêts est affiché. Ce composant est capable de communiquer avec une API externe, *MapAPI*, afin de récupérer la carte des environs.

Enfin, les requêtes formulées par l'utilisateur sont transmises au serveur grâce au *ComServer*. Celui-ci se charge également de la réception des réponses du serveur.

La partie serveur

Le composant *ComClient* réceptionne les requêtes envoyées par les clients et les transmet au *RequestManager*. Celui-ci s'occupe alors de la réalisation des différentes étapes nécessaires au traitement des requêtes.

Premièrement, le *RequestManager* récupère l'historique de l'utilisateur, demandé au *ClientManager*. Ensuite, il se chargera de récupérer les points d'intérêts des catégories demandées par l'utilisateur situés dans les environs. C'est le composant *ComFQAPI* qui est chargé de la

communication avec la base de données de Foursquare contenant les points d'intérêts et leurs informations respectives.

Lorsque le *RequestManager* a obtenu toutes les données nécessaires, celles-ci sont transmises au *Recommender*. Celui-ci se charge alors de générer la liste triée. Pour cela, le composant communique avec le *ClientManager* afin de récupérer les informations nécessaires pour déduire les utilisateurs similaires.

Si l'utilisateur a activé l'option concernant la météo, le *Recommender* demandera les conditions actuelles au composant *WeatherCom*, chargé de récupérer l'information par le biais d'une API externe.

Lorsque le client envoie les *feedbacks* au serveur, ceux-ci sont transmis au *RequestManager* qui se chargera de les communiquer au *ClientManager*. Celui-ci mettra à jour l'historique de l'utilisateur concerné et les *feedbacks* seront donc pris en compte pour les requêtes ultérieures.

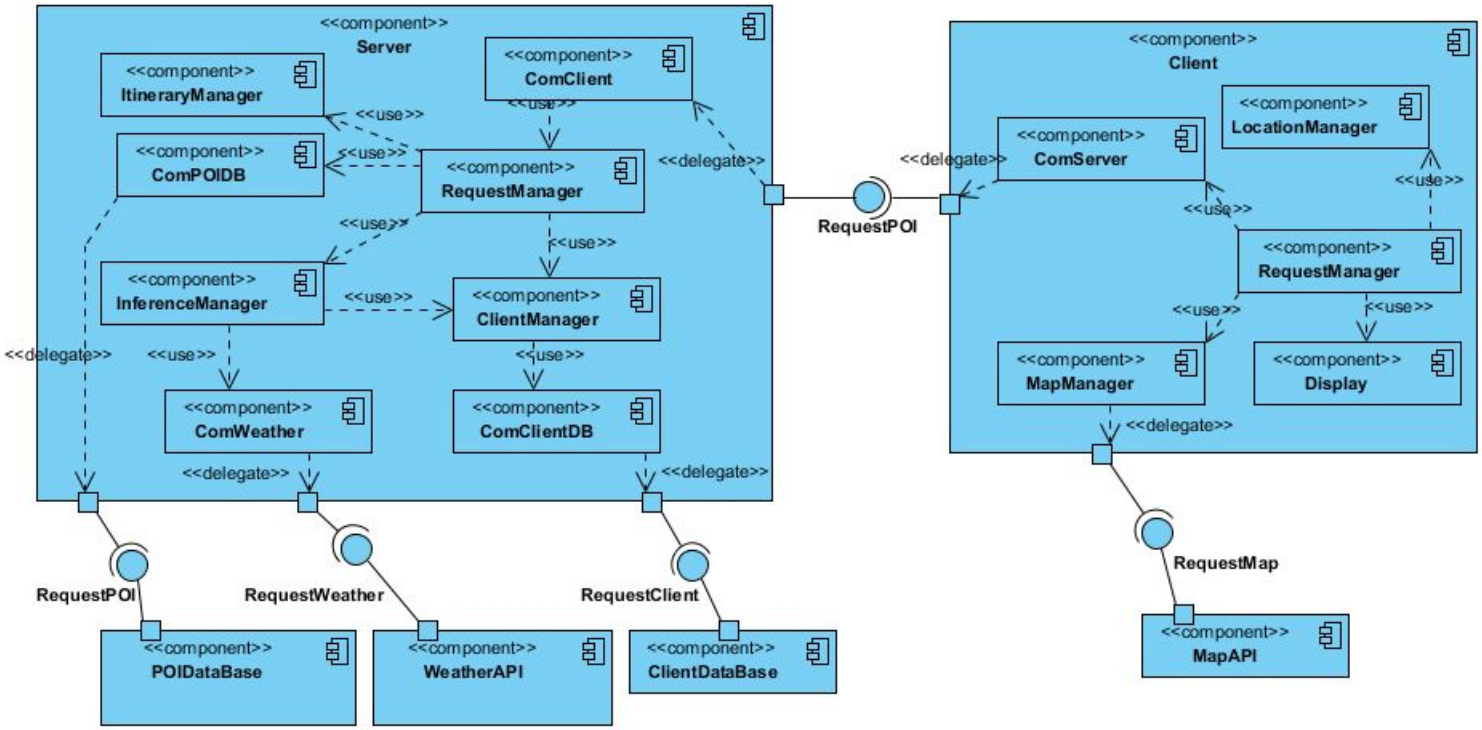

FIGURE *6.10 – Diagramme des composants*

6.5.2 Diagramme de séquence

Les figures 6.11 (p.95) et 6.12 (p.96) présentent le diagramme de séquence correspondant à la création d'un itinéraire par un utilisateur. Si celui-ci peut paraître complexe et détaillé, il a l'avantage de pouvoir présenter de manière concrète les communications existantes entre les différents composants du système.

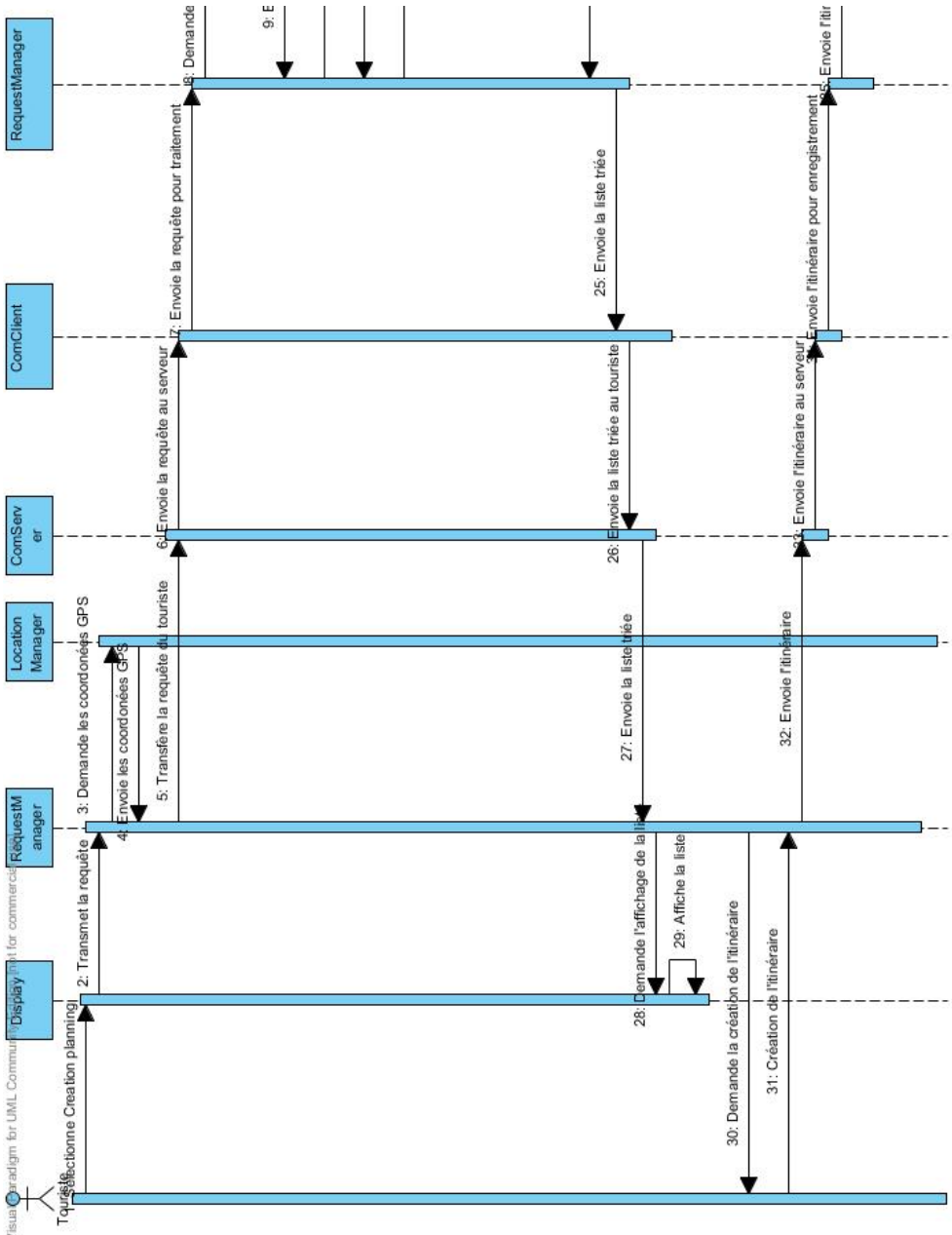

FIGURE 6.11 – *Création d'un itinéraire*

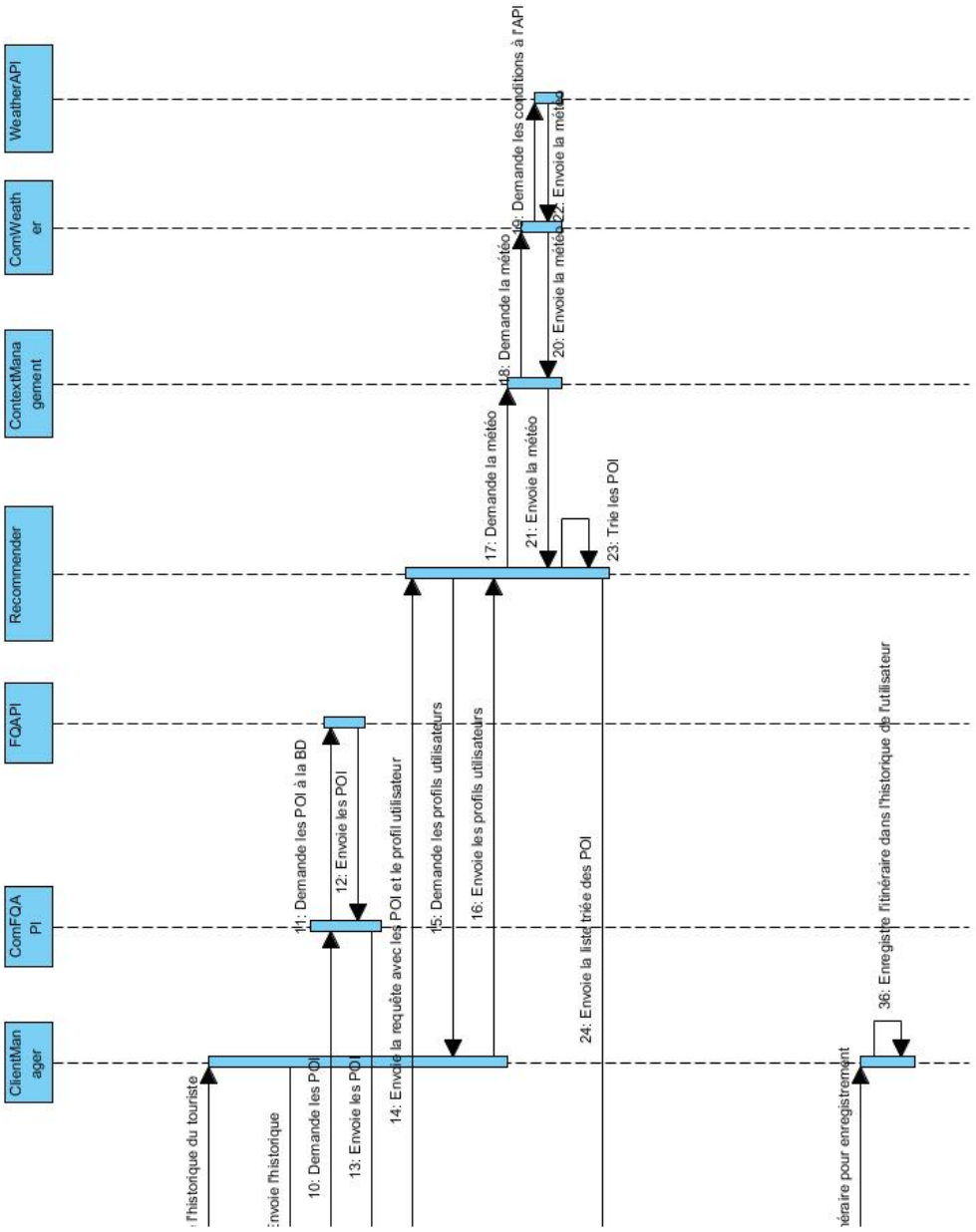

FIGURE 6.12 – *Création d'un itinéraire*

6.6 Choix technologiques

6.6.1 Java

Dès le départ, nous avons pris la décision de développer le serveur soutenant notre système dans le langage Java. Aujourd'hui, Java est fortement utilisé dans le monde informatique, et plusieurs de ses caractéristiques ont contribué à son succès.

Tout d'abord, Java est un langage interprété. Le code source est compilé en pseudo-code, interprété par une machine virtuelle, la *Java Virtual Machine* (JVM). Celle-ci agit comme une couche d'abstraction entre le matériel hôte et le code. Cet aspect permet aux programmes écrits en Java de posséder une grande portabilité, toutes les machines disposant de la JVM étant capables de les faire fonctionner de la même manière. Cette caractéristique a d'ailleurs longtemps été mise en avant par Sun, qui utilisait le slogan *Write once, run everywhere*.

Concernant le langage proprement dit, les auteurs de Java voulaient à l'époque créer un langage simple et compréhensible. Celui-ci est donc capable de gérer automatiquement la mémoire, est fortement typé et possède une syntaxe construite de manière à rester facilement lisible.

De plus, comme la majorité des langages récents, Java est orienté objet et profite des avantages associés (modulabilité, réutilisabilité...). Il est aussi multitâche et il est dès lors possible de créer plusieurs *threads* dans un même programme.

Enfin, la sécurité du système hôte est assurée de manière automatique. Un programme Java ne pouvant menacer directement le système d'exploitation et ne pouvant effectuer d'accès directs à la mémoire. Les accès aux disques durs sont aussi réglementés de manière stricte.

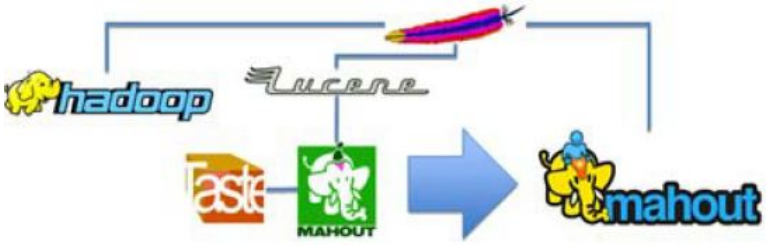

FIGURE *6.13 – Représentation des distributions de Mahout. Source : [37]*

6.6.2 Mahout

Présentation

Mahout est un *framework* de développement de solutions de *machine learning* créé par Apache [37]. La première version, numérotée 0.1, a été rendue disponible en 2008. Mahout n'était alors qu'un sous-projet issu de celui portant le nom de Lucene. Le projet Lucene consiste en un moteur de recherche *open-source*, et comporte des fonctionnalités de recherches avancées, de *text-mining* et des techniques de récupération d'information.
Certains des développeurs, appréciant le domaine de *machine learning*, ont décidé en 2010 de détacher Mahout et d'en créer un projet à part entière.

Les algorithmes de Mahout sont implémentés sur base du *framework* Java Apache Hadoop, lui-même spécialisé dans les systèmes distribués *data-intensive*. Ce *framework* possède entre autres une implémentation complète de l'algorithme *MapReduce* de Google.
Pour l'anecdote, les Mahout - aussi appelés cornacs - sont des maîtres, guides et soigneurs d'éléphants. Ce choix de nom s'explique facilement lorsque l'on sait que le logo d'Hadoop n'est autre qu'un éléphant. La maîtrise de cet animal représente donc par analogie celle d'immenses quantités de données.

L'évolution des différentes distributions du projet Mahout est représentée à la figure 6.13 (p.98) .

Actuellement, Mahout offre 4 fonctions principales :

- *Recommandation mining* : cette fonction étudie le comportement des utilisateurs et tente de proposer des objets susceptibles de les satisfaire. Ce sujet a été discuté plus en détail dans le chapitre 2 (p.9) .

- *Clustering* : les techniques de *clustering* tentent de grouper un grand nombre d'objets en plusieurs ensembles, appelés *clusters*, selon leur similarité. Ces techniques permettent également d'ordonner des ensembles de données difficilement compréhensibles et de déduire des hiérarchies. Un exemple courant de cas d'utilisation est le groupement de publications, rassemblées selon leurs sujets. Les techniques de *clustering* peuvent également se révéler efficaces pour déduire les différents segments d'une clientèle.

- *Classification* : les techniques de classification permettent de répondre aux questions telles que : « à quel point un objet appartient (ou n'appartient pas) à une catégorie ? ». À partir de catégories existantes, cette fonctionnalité peut donc apprendre quelles en sont les propriétés et ainsi assigner de nouveaux objets à leur catégorie adéquate. Aujourd'hui, ces techniques sont très fréquemment utilisées, notamment par les clients mails. En effet, chaque courriel reçu est généralement traité par ce genre d'algorithmes, afin de détecter les messages indésirables.

- *Frequent itemset mining* : prend une série de groupes d'objets et identifie quels sont ceux qui apparaissent régulièrement ensemble.

Comme discuté dans le chapitre 2 (p.9) portant sur les systèmes de recommandation, l'évolution des systèmes à travers le temps est un problème majeur. En effet, les quantités de données deviennent tellement importantes que même un ordinateur très puissant risque de ne plus être capable de les traiter correctement et de manière efficace. Les développeurs de Mahout en sont conscients. C'est pourquoi leurs choix de développements sont faits de façon à ce que le *framework* soit en mesure de gérer au mieux ce problème de données. C'est ce qui représente aujourd'hui encore l'un des avantages majeurs de Mahout sur ses concurrents.

Rappelons que nous avons fait le choix de Java pour le développement de notre système. Le fait que Mahout soit écrit et disponible dans ce langage nous a donc également confortés dans notre choix. Et si Mahout requiert l'utilisation de Java 1.6, cela ne nous pose aucun problème.

Choix de la mesure de similarité

Comme présenté au chapitre 5 (p.67) , le système utilise un algorithme de recommandation collaboratif, permettant de mettre en relation les utilisateurs. Pour cela, il est nécessaire de calculer une métrique de similarité entre ceux-ci. Si plusieurs techniques existent, nous avons décidé d'utiliser celle du *log-likelihood*. Celle-ci possède entre autres l'avantage de ne pas exiger des utilisateurs de noter les objets recommandés. Cette étape est en effet délicate (étape généralement peu appréciée par les utilisateurs, deux utilisateurs ne notent pas de la même manière...), et pouvoir l'éviter est donc intéressant.

La métrique de *log-likelihood* fait partie des techniques se basant sur les préférences de l'utilisateur, mais ignorant la valeur de cette préférence. Il n'y a donc pas de faibles ou de fortes préférences. Cette technique est inspirée du coefficient de Tanimoto correspondant au ratio entre l'union de l'ensemble total des objets préférés par deux utilisateurs et l'intersection des objets préférés en commun. Le schéma de la figure 6.14 (p.101) illustre ce coefficient.

Si le *log-likelihood* repose également sur l'intersection des préférences de deux utilisateurs, il tente de mesurer à quel point obtenir une telle intersection est improbable, étant donné le nombre total d'objets qui ont été préférés par ces deux utilisateurs individuellement.

Illustrons ce procédé par un exemple : dans le cas où deux touristes ont visité la tour Eiffel, peut-on les considérer comme similaires ? Si nos deux touristes ont chacun visité des centaines d'endroits à travers le globe, et que le seul point commun entre eux est la visite de la tour Eiffel, il est très peu probable qu'ils le soient. De très nombreux touristes visitent cette attraction. En

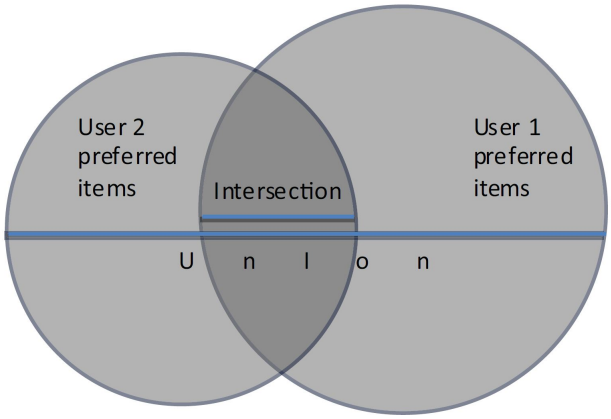

FIGURE *6.14 – Le coefficient de Tanimoto est le ratio entre l'intersection et l'union des préférences de deux utilisateurs. Source : [37]*

revanche, si nos deux vacanciers ne possèdent chacun que quelques points d'intérêts visités dans leur historique, on pourra les considérer comme similaires.

Si le coefficient de Tanimoto reprend déjà cette idée en calculant le ratio, le *log-likelihood* tente d'aller plus loin en calculant, par des tests statistiques, à quel point il est improbable que l'intersection de deux utilisateurs soit dûe à la chance. Plus il est improbable que deux utilisateurs ne possèdent aucune ressemblance dans leurs préférences, plus la valeur de similarité sera élevée.

Grâce à cela, le *log-likelihood* est considéré comme plus intelligent que le coefficient de Tanimoto, et produira généralement des résultats plus précis [37].

6.6.3 Foursquare

SmartTrip utilise Foursquare uniquement afin d'obtenir une base de données complète et entretenue des points d'intérêts. Cela permet notamment d'avoir une application fonctionnelle dans toutes les régions du globe. Les services plus spécifiques au réseau social (attributions de badges, commentaires...) disponibles pour les applications tierces seront donc ignorés.

Foursquare est un réseau social créé en 2009 par Dennis Crowley et Naveen Selvadurai. Ces créateurs sont à l'origine de Dodgeball, rachetée par Google en 2005 et qui a donné naissance à Google Latitude.

Foursquare est aujourd'hui disponible sur les différentes plateformes mobiles et permet aux utilisateurs de découvrir des points d'intérêts situés à proximité. Une fois sur place, l'utilisateur peut réaliser un « check-in », et ainsi déclarer sa position à son réseau social. Un aspect ludique y est intégré, car l'utilisateur ayant réalisé le plus de « check-in » dans un endroit en est déclaré « Maire ».

Depuis peu, Foursquare tente de monétiser son service, en proposant aux entreprises de payer pour que leurs produits soient recommandés en priorité aux utilisateurs du réseau.

L'évolution du réseau social *location-based* est impressionnante, passant de 5 à 15 millions d'utilisateurs. À la fin de l'année 2011, la moitié des utilisateurs étaient américains. Néanmoins, les développeurs souhaitent améliorer leur présence dans le reste du monde : depuis septembre 2011, Foursquare est disponible en 11 langues différentes. Foursquare a donc réussi à se faire une place dans le monde très compétitif des réseaux sociaux. Sa présence de plus en plus prononcée sur le numéro 1 des réseaux, Facebook, permet de lui garantir un futur prospère. Si SmartTrip n'utilise aucune fonctionnalité spécifique à ce réseau social, la croissance de Foursquare garantit une base de données complète et à jour des points d'intérêts présents dans le globe.

De plus, une API complète et correctement documentée est disponible pour les développeurs.

Les catégories de points d'intérêts proposées par Foursquare étant différentes de celles présentes dans SmartTrip, une table de correspondances a dû être effectuée et se présente comme indiqué au tableau 6.1 (p.103) .

SmartTrip	Foursquare
Nature (Non recommandé par mauvais temps)	Beach, Field, Garden, Hiking Trail, Lake, Park, Rivers, Scenic Lookout, Vineyard
Nature	Campground, Garden Centers, Hot Spring, Mountains, Zoo or Aquariums
Party	Bars, Beer Gardens, Cocktails Bars, Dive Bars, Gay Bars, Hookah Bars, Hotel Bars, Karaoke Bars, Lounges, Music Venues, Nightclubs, Other Nightlife, Pubs, Sake Bars, Speakeasies, Sports Bars, Strip Clubs, Whisky Bars, Wine Bars
Sport (extérieur)	Baseball Stadiums, Cricket Grounds, Football Stadiums, Soccer Stadiums, Tracks, Racetracks, Baseball Fields, Surf Spots, Skate Parks, Skating Rinks, Soccer Fields
Sport (intérieur)	Basketball Stadiums, Hockey Stadiums, Pool Halls, Bowling Alleys, Basketball Courts, Golf Courses, Pools
Culture	Art Galleries, Comedy Clubs, Indie Movie Theaters, Multiplexes, Art Museums, History Museums, Museums, Planetariums, Science Museums, Concert Halls, Jazz Clubs, Piano Bars, Rock Clubs, Concert Halls, Dance Studios, Indie or Off Broadway Theaters, Opera Houses, Theaters, Churches, Mosques, Synagogues, Temples, Plazas
Gastronomy	Food, Drinks, Coffee
Shopping	Shops

TABLE *6.1 – Table des correspondances des catégories d'activités*

6.6.4 Google Weather

Nous l'avons précisé à plusieurs reprises, SmartTrip doit être capable de connaître les conditions météorologiques de la région dans laquelle l'utilisateur se trouve.

Si de nombreuses API existent, la « Weather API » de Google fournit exactement les informations dont nous avons besoin. En effet, beaucoup de services se contentent d'envoyer les informations « brutes » concernant la température, la force du vent ou encore le taux d'humidité. Si ces données sont précises, elles n'en sont pas utiles pour autant. Comment peut-on, à partir de ces données uniquement, savoir si les conditions météo actuelles permettent la visite d'attractions extérieures ?

Google possède l'avantage de fournir en plus les conditions météo sous la forme d'un texte et « raining », « snowy », « sunny »... sont des réponses que l'on peut obtenir. Si celles-ci sont moins précises, elles sont néanmoins interprétables beaucoup plus facilement.

Par ailleurs, l'API est tolérante quant au format des données d'entrée et utilise le XML, facile d'utilisation, lors des réponses aux requêtes. Un exemple de réponse est présenté en annexe A (p.111) .

Google étant aujourd'hui l'un des géants du web, l'entreprise californienne n'est plus à présenter. Néanmoins, la bonne réputation de la compagnie n'a pu que nous influencer positivement dans notre choix, assurant une bonne stabilité de l'API et des résultats fiables.

6.6.5 Android

À l'origine, Android est une *start-up* créée en 2003 par Andy Rubin, un ingénieur américain à l'origine du téléphone/PDA HipTop Danger. Si ce modèle est peu connu en Europe, il fut populaire aux États-Unis et a détenu à l'époque jusqu'à 6% du traffic e-mail mobile.

Durant l'année 2005, Google annonçait le rachat d'Android, alors peu connu. Jusqu'en 2008, Android consistait en un système d'exploitation *open source* à destination des appareils mobiles. Le logiciel s'appelait alors gPhone et était proposé gratuitement aux fabricants d'appareils mobiles.
L'acquisition d'Android par Google a notamment permis aux actions du géant américain d'atteindre un niveau historique de 524 dollars le 5 novembre 2007.

Aujourd'hui, le système d'exploitation est fondé sur un noyau Linux, et le développement des applications se fait dans une version allégée de Java. L'enregistrement des données repose sur SQLite et est conçu de manière à faciliter le fonctionnement des autres applications de Google, telles que Gmail, Google Maps...

Android est désormais l'un des leaders des systèmes d'exploitation mobiles. Sur le premier trimestre de 2011, 35,7 millions de smartphones embarquant le système d'exploitation Android ont été vendus. Cela représente une part de marché de 35%. Depuis 2010, Android est rentable pour Google, qui a estimé un gain de plus d'un milliard de dollars à la fin de la même année.

L'environnement Android a été choisi pour le développement de la partie mobile du système, et ce pour des raisons essentiellement pratiques.

En effet, Android est aujourd'hui disponible sur un grand nombre d'appareils mobiles, tant au niveau des téléphones que des tablettes. Développer pour cet environnement permet donc de viser un plus large public. De plus, le langage utilisé pour le développement des applications est une version adaptée de Java. Cela permet dès lors d'éviter l'apprentissage de langages propriétaires.
Un émulateur à destination des développeurs et régulièrement mis à jour est aussi disponible, ce qui rend le développement plus rapide.

Enfin, le succès d'Android a conduit à une communauté assez importante de développeurs sur le web. Il est dès lors aisé de trouver des informations et des réponses aux problèmes rencontrés.

De plus, les chercheurs et petits développeurs ont généralement une préférence au système de Google par rapport à l'iOS de l'Apple. Ce sujet a été abordé au début de ce mémoire 1.1.2 (p.3)
.

Chapitre 7

Conclusion et Perspectives

7.1 Conclusion

Avec la personnalisation du web, les techniques de recommandation et de contextualisation sont plus que jamais à l'ordre du jour. Ainsi, de plus en plus de systèmes utilisent ces techniques afin de proposer des résultats les plus pertinents possible à leurs utilisateurs. Le milieu d'application le plus connu étant celui de la publicité. Depuis quelques années, de nombreuses recherches ont été effectuées afin de concilier les techniques de contextualisation et de recommandation dans le domaine touristique, sans arriver à faire l'unanimité. C'est dans ce domaine que s'inscrit ce travail, l'objectif étant de proposer un système de recommandation contextualisé de points d'intérêts touristiques. Pour ce faire, le document est constitué de plusieurs étapes.

Dans un premier temps, dans le chapitre 2 (p.9) , nous avons introduit les paradigmes de la contextualisation, de la recommandation et de la recommandation contextualisée. Pour chacun, une définition a été proposée et les différentes techniques ont été détaillées. Nous avons également passé en revue leurs inconvénients respectifs. Dans la suite de ce chapitre, nous avons présenté les ontologies, les langages qui y sont associés et les moteurs d'inférences existants. Enfin, nous avons détaillé la composition générale d'une ontologie et introduit leur utilisation comme support

de contexte.

Dans la suite de ce travail, nous avons parcouru dans le chapitre 3 (p.33) les recherches et projets principaux réalisés dans le domaine au cours de ces dernières années. Pour chacun, les avantages et inconvénients ont été mis en évidence. Enfin, une synthèse relevant les problèmes récurrents rencontrés par les auteurs a été réalisée.

Dans le chapitre 4 (p.59) , modélisation des utilisateurs potentiels, nous avons recueilli les comportements typiques des touristes. Nous avons pu constater que ceux-ci étaient confrontés à une suite de problèmes (quoi, comment, quand, où, partage) dont la résolution ne devait cependant pas être vue comme une corvée. Nous avons également présenté les outils traditionnels couramment utilisés lors des séjours touristiques et abordé la manière dont une application mobile pouvait s'y substituer, ou du moins compléter ces outils. Toutes ces observations avaient pu être réalisées sur base de plusieurs études, dont celle de Brown et Chalmers [17]. Ce chapitre s'est clôturé en présentant les différentes possibilités qu'offrait la technologie au sein du monde du tourisme.

Ensuite, à travers le chapitre 5 (p.67) , nous avons décrit l'analyse et l'architecture de notre système de recommandation touristique contextualisé. Pour cela, nous avons commencé par rappeler les problèmes rencontrés par les touristes et la manière de les résoudre grâce à un système de recommandation contextualisé.

Nous avons ensuite présenté une ontologie permettant de représenter le contexte d'une visite effectuée par un utilisateur du système. Lors de la conception de celle-ci, et puisque les ontologies sont construites de manière à être réutilisables, nous avons choisi comme base l'ontologie SOUPA (présentée au chapitre 3 (p.33)). Après avoir construit notre ontologie, nous avons abordé la recommandation contextualisée. Dans cette partie, nous avons rappelé les caractéristiques de la technique de *pre-filtering* et construit des règles de filtrage contextuel. Nous avons ensuite justifié notre utilisation d'un système de recommandation bidimensionnel de type collaboratif.

Ce chapitre se termine en présentant l'architecture globale du système touristique, séparée en

3 parties principales : la partie serveur, la partie client et la partie service. Nous avons également rappelé les problèmes récurrents présentés dans le chapitre 3 (p.33) et les solutions que nous y avions apportées lors de la conception de notre architecture.

Enfin, nous avons présenté au chapitre 6 (p.81) le prototype développé, servant de validation technologique à la solution théorique proposée précédemment. Nous avons d'abord énoncé les objectifs et les exigences que le prototype se devait de remplir.

Ensuite, nous avons présenté et illustré au moyen de captures d'écran les différentes étapes devant être effectuées lors de l'utilisation du prototype pour la création d'un itinéraire touristique. Nous avons également détaillé l'architecture du système par le biais d'un diagramme de composants et d'un diagramme de séquence.

Enfin, nous avons clôturé ce chapitre en présentant et en justifiant les choix technologiques qui ont pu être effectués lors du développement du prototype.

7.2 Perspectives

Après avoir réalisé ce travail, nous constatons que plusieurs pistes de recherches intéressantes existent.

Tout d'abord, il serait intéressant d'effectuer une étude comparative plus poussée entre les différents paradigmes de recommandation contextualisée (présentés à la section 2.4 (p.20)). En effet, si nous savons que de manière générale, le *pre-filtering* s'avère moins coûteux, nous savons également que le choix peut dépendre fortement du type d'application développée (cf. [8]). Une comparaison approfondie des trois paradigmes permettrait de connaître les situations dans lesquelles ils sont les plus efficaces.

Concernant les systèmes de recommandation traditionnels, nous n'avons pas détaillé et comparé les possibilités de calcul de similarité. En effet, seuls le coefficient de Tanimoto et le *log likelihood*, n'exigeant pas de *feedbacks* de valeur de préférences de la part des utilisateurs, ont été

abordés à la section 6.6.2 (p.98) . D'autres techniques existent et les analyser pourrait déterminer avec précision laquelle est la plus adaptée ici.

Tout au long de ce travail, nous nous sommes donc concentrés sur les techniques de recommandation et de contextualisation appliquées au domaine touristique. Dans notre cas, les points d'intérêts sont fournis par une base de données, pouvant être interne ou externe au système. Dans la continuité de ce travail, nous pourrions étudier les possibilités pour connecter des *web services* au système. Ces *web services* pourraient aussi bien fournir des informations de contexte que des points d'intérêts. Pour cela, il est nécessaire d'étudier les technologies associées (BPEL, technologies XML...) et les possibilités d'intégration dans l'architecture existante.

Le chapitre 4 (p.59) se concentre essentiellement sur les comportements globaux de touristes et la manière dont ils utilisent certains outils. Nous n'avons cependant pas relevé les typologies de touristes existantes. Une étude future pourrait tenter de les mettre en avant et d'analyser les possibilités d'adaptation des systèmes de recommandation et de contextualisation selon ces typologies.

Une autre évolution attendue serait le consensus de la communauté scientifique à propos d'une ontologie de contexte standard. En effet, comme l'a souligné Ay [13], l'apport le plus important des ontologies est l'interopérabilité, mais cette qualité ne transparaîtra que quand il existera un standard largement accepté.

Concernant le prototype, il serait utile d'en réaliser une validation en conditions réelles. Récolter les *feedbacks* de différents types de touristes possédant des comportements divers permettrait de connaître leurs ressentis et de s'assurer que les problèmes récurrents vus à la section 3.11 (p.55) ont pu être évités.

Annexe A

Réponse de l'API Google Weather

Listing A.1 – Exemple de réponse XML de l'API

```
<xml_api_reply version="1">
<weather module_id="0" tab_id="0" mobile_row="0"
mobile_zipped="1" row="0" section="0">
<forecast_information>
<city data="Porto Alegre, Rio Grande do Sul"/>
<postal_code data="Porto Alegre"/>
<latitude_e6 data=""/>
<longitude_e6 data=""/>
<forecast_date data="2012-07-30"/>
<current_date_time data="1970-01-01 00:00:00 +0000"/>
<unit_system data="SI"/>
</forecast_information>
<current_conditions>
```

```
<condition data="Couverture nuageuse partielle"/>
<temp_f data="52"/>
<temp_c data="11"/>
<humidity data="Humidité : 82 %"/>
<icon data="/ig/images/weather/partly_cloudy.gif"/>
<wind_condition data="Vent : E à 6 km/h"/>
</current_conditions>
<forecast_conditions>
<day_of_week data="lun."/>
<low data="9"/>
<high data="17"/>
<icon data="/ig/images/weather/mostly_sunny.gif"/>
<condition data="Ensoleillé dans l'ensemble"/>
</forecast_conditions>
<forecast_conditions>
<day_of_week data="mar."/>
<low data="12"/>
<high data="17"/>
<icon data="/ig/images/weather/chance_of_rain.gif"/>
<condition data="Risques de pluie"/>
</forecast_conditions>
<forecast_conditions>
<day_of_week data="mer."/>
<low data="15"/>
<high data="26"/>
<icon data="/ig/images/weather/mostly_sunny.gif"/>
<condition data="Partiellement ensoleillé"/>
</forecast_conditions>
<forecast_conditions>
```

```
<day_of_week data="jeu."/>
<low data="15"/>
<high data="26"/>
<icon data="/ig/images/weather/mostly_sunny.gif"/>
<condition data="Ensoleillé dans l'ensemble"/>
</forecast_conditions>
</weather>
</xml_api_reply>
```

Bibliographie

[1] inference - definition of inference by the free online dictionary, thesaurus and encyclopedia., . URL http://www.thefreedictionary.com/inference.

[2] Ontologie : Définition de ontologie., . URL http://www.cnrtl.fr/definition/ontologie.

[3] Guides papier ou guides numériques ? URL http://voyage-californie.be/guides-papier-ou-guides-web.

[4] Android devant apple dépasse les 50% du marché de marché des smartphones aux etats-unis en février 2012 (comscore). URL http://www.eco-conscient.com/art-969-part-de-marche-android-depasse-apple-sur-le-segment-des-smartphones-android-appl html.

[5] Tourisme international : les premiers résultats pour lŠannée 2011 confirment la consolidation de la croissance. URL http://media.unwto.org/fr/press-release/2011-05-11/tourisme-international-les-premiers-resultats-pour-l-annee-2011-confirmen-0.

[6] Profile of overseas travelers to the united states : 2011. URL http://www.tinet.ita.doc.gov/outreachpages/download_data_table/2011_Overseas_Visitor_Profile.pdf.

[7] Gregory D. Abowd, Christopher G. Atkeson, Jason Hong, Sue Long, Rob Kooper, and Mike Pinkerton. Cyberguide : a mobile context-aware tour guide. *Wirel. Netw.*, 3(5) : 421–433, October 1997. ISSN 1022-0038. doi : 10.1023/A:1019194325861. URL http://dx.doi.org/10.1023/A:1019194325861.

[8] Gediminas Adomavicius and Alexander Tuzhilin. Context-aware recommender systems. In *Proceedings of the 2008 ACM conference on Recommender systems*, RecSys '08, pages 335–336, New York, NY, USA, 2008. ACM. ISBN 978-1-60558-093-7. doi : 10.1145/1454008.1454068. URL `http://doi.acm.org/10.1145/1454008.1454068`.

[9] Gediminas Adomavicius, Ramesh Sankaranarayanan, Shahana Sen, and Alexander Tuzhilin. Incorporating contextual information in recommender systems using a multidimensional approach. *ACM Trans. Inf. Syst.*, 23(1) :103–145, January 2005. ISSN 1046-8188. doi : 10.1145/1055709.1055714. URL `http://doi.acm.org/10.1145/1055709.1055714`.

[10] Xavier Amatriain. Recommender systems : We're doing it (all) wrong. URL `http://technocalifornia.blogspot.be/2011/04/recommender-systems-were-doing-it-all.html`.

[11] Sarabjot Anand and Bamshad Mobasher. Intelligent Techniques for Web Personalization. pages 1–36. 2005. doi : 10.1007/11577935_1. URL `http://dx.doi.org/10.1007/11577935_1`.

[12] Grigoris Antoniou, Enrico Franconi, and Frank Van Harmelen. Introduction to semantic web ontology languages. In *Reasoning Web, Proceedings of the Summer School, Malta, 2005. Number 3564 in Lecture Notes in Computer Science*. Springer, 2005.

[13] F. Ay. Context Modeling and Reasoning using Ontologies. URL `http://www.ponnuki.de/cmaruo/cmaruo.pdf`.

[14] Marko Balabanović and Yoav Shoham. Fab : content-based, collaborative recommendation. *Commun. ACM*, 40(3) :66–72, March 1997. ISSN 0001-0782. doi : 10.1145/245108.245124. URL `http://doi.acm.org/10.1145/245108.245124`.

[15] N. Baumgartner and W. Retschitzegger. A survey of upper ontologies for situation awareness. *Proc. of the 4th IASTED International Conference on Knowledge Sharing and Collaborative Engineering, St. Thomas, US VI*, pages 1Ű9+, 2006.

[16] François Bostnavaron. Le guide touristique papier fait de la résistance. URL `http://www.lemonde.fr/style/article/2012/07/23/le-guide-papier-fait-de-la-resistance_1736774_1575563.html`.

[17] Barry Brown and Matthew Chalmers. Tourism and mobile technology. In *ECSCW'03 : Proceedings of the eighth conference on European Conference on Computer Supported Co-operative Work*, pages 335–354, Norwell, MA, USA, 2003. Kluwer Academic Publishers. URL `http://portal.acm.org/citation.cfm?id=1241907&dl=GUIDE&coll=GUIDE&CFID=21538253&CFTOKEN=61307056`.

[18] L. Buriano. Exploiting Social Context Information in Context-Aware Mobile Tourism Guides. In *Proceedings of Mobile Guide'06*, 2006.

[19] John M. Carroll and Mary Beth Rosson. Interfacing thought : cognitive aspects of human-computer interaction. chapter Paradox of the active user, pages 80–111. MIT Press, Cambridge, MA, USA, 1987. ISBN 0-262-03125-6. URL `http://dl.acm.org/citation.cfm?id=28446.28451`.

[20] Federica Cena, Luca Console, Cristina Gena, Anna Goy, Guido Levi, Sonia Modeo, and Ilaria Torre. Integrating heterogeneous adaptation techniques to build a flexible and usable mobile tourist guide. *AI Commun.*, 19(4) :369–384, December 2006. ISSN 0921-7126. URL `http://dl.acm.org/citation.cfm?id=1219755.1219764`.

[21] Harry Chen, Filip Perich, Tim Finin, and Anupam Joshi. SOUPA : Standard Ontology for Ubiquitous and Pervasive Applications. *Mobile and Ubiquitous Systems, Annual International Conference on*, 0 :258–267, 2004. doi : 10.1109/MOBIQ.2004.1331732. URL `http://dx.doi.org/10.1109/MOBIQ.2004.1331732`.

[22] Harry Coccossis and Mary Constantoglou. The use of typologies in tourism planning : Problems and conflicts. ERSA conference papers ersa06p712, European Regional Science Association, August 2006. URL `http://ideas.repec.org/p/wiw/wiwrsa/ersa06p712.html`.

[23] Anind K. Dey. Understanding and using context. *Personal Ubiquitous Comput.*, 5(1) :4–7, January 2001. ISSN 1617-4909. doi : 10.1007/s007790170019. URL `http://dx.doi.org/10.1007/s007790170019`.

[24] Chiel Drost. Privacy in context-aware systems. URL `http://www.utwente.nl/ewi/asna/assignments/completed/drost.pdf`.

[25] Natalie Fedorowicz. Tourisme international 2010 : une reprise à plusieurs vitesses. URL `http://www.ced.travel/fr/nouvelles-des-destinations/132-international-tourism-2010-multi-speed-recovery.html`.

[26] V. Haarslev and R. Möller. Racer : A core inference engine for the semantic web. In *Proceedings of the 2nd International Workshop on Evaluation of Ontology-based Tools (EON2003), located at the 2nd International Semantic Web Conference ISWC 2003, Sanibel Island, Florida, USA, October 20*, pages 27–36, 2003.

[27] Ian Horrocks. Ontologies and the semantic web. *Communications of the ACM*, 51(12) : 58–67, December 2008. URL `download/2008/Horr08a.pdf`.

[28] Xavier Lacot. Introduction à owl, un langage xml d'ontologies web. June 2005. URL `http://lacot.org/public/introduction_a_owl.pdf`.

[29] Carlos Lamsfus, Aurkene Alzua-Sorzabal, David Martin, Zigor Salvador, and Alex Usandizaga. Human-centric ontology-based context modelling in tourism. In *KEOD*, pages 424–434, 2009.

[30] Carlos Lamsfus, Aurkene Alzua-Sorzabal, David Martin, and Zigor Salvador. Semantic-based contextual computing support for human mobility. In Ulrike Gretzel, Rob Law, and Matthias Fuchs, editors, *ENTER*, pages 603–615. Springer Vienna, 2010. ISBN 978-3-211-99406-1. URL `http://dblp.uni-trier.de/db/conf/enter/enter2010.html#LamsfusAMS10`.

[31] Phillip Lord. Components of an ontology, 2010. URL `http://ontogenesis.knowledgeblog.org/514`.

[32] Gaëlle Lucas. Espagne : le tourisme ne mettra pas la crise en vacances. URL `http://www.latribune.fr/actualites/economie/union-europeenne/20120705trib000707524/espagne-le-tourisme-ne-mettra-pas-la-crise-en-vacances.html`.

[33] RICHARD MACMANUS. 5 problems of recommender systems., 2009. URL `http://www.readwriteweb.com/archives/5_problems_of_recommender_systems.php`.

[34] Miki Nakagawa and Bamshad Mobasher. A Hybrid Web Personalization Model Based on Site Connectivity. URL `http://www.csie.ncue.edu.tw/~{}myshih/61021/NM03b.pdf`.

[35] Kenta Oku, Shinsuke Nakajima, Jun Miyazaki, and Shunsuke Uemura. Context-aware svm for context-dependent information recommendation. In *Proceedings of the 7th International Conference on Mobile Data Management*, MDM '06, pages 109–, Washington, DC, USA, 2006. IEEE Computer Society. ISBN 0-7695-2526-1. doi : 10.1109/MDM.2006.56. URL `http://dx.doi.org/10.1109/MDM.2006.56`.

[36] Michael P. OŠMahony, Neil J. Hurley, and Guénolé C. M. Silvestre. Recommender systems : Attack types and strategies. In *IN PROCEEDINGS OF THE 20TH NATIONAL CONFERENCE ON ARTIFICIAL INTELLIGENCE (AAAI-05*, pages 334–339. AAAI Press, 2005.

[37] Sean Owen, Robin Anil, Ted Dunning, and Ellen Friedman. *Mahout in Action*. Manning Publications, 1 edition, January 2011. ISBN 1935182684. URL `http://www.amazon.com/exec/obidos/redirect?tag=citeulike07-20&path=ASIN/1935182684`.

[38] Umberto Panniello, Alexander Tuzhilin, Michele Gorgoglione, Cosimo Palmisano, and Anto Pedone. Experimental comparison of pre- vs. post-filtering approaches in context-aware recommender systems. In *Proceedings of the third ACM conference on Recommender systems*, RecSys '09, pages 265–268, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-435-5. doi : 10.1145/1639714.1639764. URL `http://doi.acm.org/10.1145/1639714.1639764`.

[39] S Saeedi, N El-Sheimy, M R Malek, and N N Samany. An ontology based context modelling approach for mobile touring and navigation system. *Canadian Geomatics Conference and Symposium of Commission I ISPRS*, 2010.

[40] Andrew I. Schein, Alexandrin Popescul, Lyle H. Ungar, and David M. Pennock. Methods and metrics for cold-start recommendations. In *Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '02, pages 253–260, New York, NY, USA, 2002. ACM. ISBN 1-58113-561-0. doi : 10.1145/564376.564421. URL `http://doi.acm.org/10.1145/564376.564421`.

[41] Bill N. Schilit, Norman Adams, and Roy Want. Context-aware computing applications. In *IN PROCEEDINGS OF THE WORKSHOP ON MOBILE COMPUTING SYSTEMS AND APPLICATIONS*, pages 85–90. IEEE Computer Society, 1994.

[42] Thomas Strang and Claudia Linnhoff-Popien. A context modeling survey. In *In : Workshop on Advanced Context Modelling, Reasoning and Management, UbiComp 2004 - The Sixth International Conference on Ubiquitous Computing, Nottingham/England*, 2004.

[43] Mark van Setten, Stanislav Pokraev, and Johan Koolwaaij. Context-aware recommendations in the mobile tourist application compass. *Adaptive Hypermedia and Adaptive Web-Based Systems*, pages 235–244, 2004. URL `http://www.springerlink.com/content/yk2m700k15kmdej2`.

[44] Xiao Hang Wang, Da Qing Zhang, Tao Gu, and Hung Keng Pung. Ontology based context modeling and reasoning using owl. *Pervasive Computing and Communications Workshops, IEEE International Conference on*, 0 :18, 2004. doi : 10.1109/PERCOMW.2004.1276898. URL `http://www.computer.org/portal/web/csdl/doi/10.1109/PERCOMW.2004.1276898`.

[45] Norbert Weienberg, Agns Voisard, and Rdiger Gartmann. Using ontologies in personalized mobile applications. In *Proceedings of the 12th annual ACM international workshop on Geographic information systems*, GIS '04, pages 2–11, New York, NY, USA, 2004. ACM. ISBN 1-58113-979-9. doi : http://doi.acm.org/10.1145/1032222.1032225. URL `http://doi.acm.org/10.1145/1032222.1032225`.

[46] Juan Ye, Lorcan Coyle, Simon Dobson, and Paddy Nixon. Ontology-based models in pervasive computing systems. *Knowl. Eng. Rev.*, 22(4) :315–347, December 2007.

ISSN 0269-8889. doi : 10.1017/S0269888907001208. URL `http://dx.doi.org/10.1017/S0269888907001208`.

[47] Chien-Chih Yu and Hsiao ping Chang. Personalized location-based recommendation services for tour planning in mobile tourism applications. In Tommaso Di Noia and Francesco Buccafurri, editors, *EC-Web*, volume 5692 of *Lecture Notes in Computer Science*, pages 38–49. Springer, 2009. ISBN 978-3-642-03963-8. URL `http://dblp.uni-trier.de/db/conf/ecweb/ecweb2009.html#YuC09`.

[48] Zhiwen Yu, Xingshe Zhou, Daqing Zhang, Chung-Yau Chin, Xiaohang Wang, and Ji Men. Supporting context-aware media recommendations for smart phones. *IEEE Pervasive Computing*, 5 :68–75, 2006. ISSN 1536-1268. doi : http://doi.ieeecomputersociety.org/10.1109/MPRV.2006.61.